LinkEHR Editor/Studio

User's Manual

(v20131012)

Copyright © 2005-2013

Universitat Politècnica de València

www.linkehr.com
www.ibime.upv.es

LinkEHR® is a Registered Trademark

Table of Content

1. About this manual 4
2. What is LinkEHR? 4
2. Installing LinkEHR 6
3. Archetype Edition: step by step tutorial 7
   4.1 Running LinkEHR for the first time 7
   4.2 Configure LinkEHR 8
   4.3 Adding a new reference model to LinkEHR 11
   4.4 Creating a new archetype 16
   4.5 Opening an archetype 17
   4.6 Editing an archetype 17
   4.7 Specializing an archetype 26
   4.8 Validating an archetype 26
   4.9 Save an archetype 26
   4.10 Reference model specific editors 27
   4.11 Defining a reference model documentation 27
   4.12 Translating archetypes to another language 29
5. Mapping archetypes to data sources 30
   5.1 Comprehensive archetypes 30
   5.2 Mapping of archetypes to data sources 32
   5.3 Integration Archetype Edition: step by step tutorial 33
      5.3.1 Data source import 35
      5.3.2 Attribute mapping definition 36
      5.3.3 Object builder definition 45
      5.3.4 Generation of covers and XQuery transformation scripts 46
6. Customized archetype editor for CEN EN13606 50
   6.1 Choosing a root entity for a CEN EN13606 archetype 50
   6.2 Editing CEN EN13606 archetypes with LinkEHR 52
      6.2.1 CEN EN13606 Datatypes 53
      6.2.2 Creating archetype internal references and archetype slots 54
   6.3 Editing HL7 CDA archetypes with LinkEHR 57
      6.3.1 Choosing a root entity for a HL7 CDA archetype 57
      6.3.2 Editing HL7 CDA archetypes with LinkEHR 58
      6.3.3 HL7 CDA Datatypes 60

1. About this manual

This manual provides a brief description of the main functionalities of both LinkEHR Editor and LinkEHR Studio. Severeal of those functionalities are only available in the commercial versions of the tool. In this document, those features will be marked with the following icon:

COM

To get the full versions of LinkEHR Editor/Studio, please contact us.

2. What is LinkEHR?

LinkEHR is a visual tool implemented in Java under the Eclipse platform which allows the edition of archetypes based on different reference models, the specification of mappings between archetypes and data sources and the semi-automatic generation of data conversion scripts which translate not normalized data into XML documents which conform to the reference model and at the same time satisfy the data constraints imposed by archetypes.

LinkEHR explores the use of archetypes as a means to achieve standardization and semantic data integration of distributed health data. The main objectives are twofold.

Firstly, in the context of data integration, to use of archetypes as a semantic layer over the data repositories, whose contents need to be integrated and exchanged, associating them with formal semantics. As aforementioned, the main purpose of archetypes is to describe information in the form of a set of machine-computable domain concept definitions based on a reference model. Therefore, archetypes may act as semantic descriptions that Figure the information contents of heterogeneous repositories.

Secondly, we intend to employ archetypes for making public existing clinical information in the form of standardized EHR extracts. For this purpose, we take advantage of their data definition facet, which was formalized in the previous section. Archetypes explicitly specify the structure and content of valid instances of the underlying reference model to which interchangeable data instances must conform. Thus, it becomes necessary to transform data from the local sources with a particular structure or schema to meet the data structures defined by archetypes. This problem is known in the literature as the data exchange (translation or transformation) problem. Data exchange requires at the schema level an explicit representation of how the source and target schemas are related to each other; these explicit representations are called mappings. Mappings between two schemas can be specified in different ways. They can be a query expressed in SQL or XQuery, a predicate in first order logic, a set of correspondences each of which relates a element from the source schema to an element from the target schema or even a third schema that relates the other two via two sets of correspondences, i.e. a reified mapping. At the data level, data transformations operate on the instances of the schemas rather than on the schemas

themselves, they transform source instances into target instances according to the mapping defined between the corresponding schemas. Data transformations may be expressed in a specific data transformation language, such as XQuery or by using a general purpose language as Java.

Since the health data to be made public resides in the underlying data sources, it is necessary to define some kind of mapping information that links entities described in the archetype (object nodes and attributes) to data elements in data repositories (e.g. elements and attributes in the case of XML documents, tables and attributes in the case of relational data sources). We use the term integration archetype to denote an archetype for which a mapping specification to a set of data sources has been defined, i.e.:

Integration archetype = archetype + mapping specification.

An integration archetype can be considered to be a view that provides abstraction in interfacing between the data sources that hold the data to be shared and the reference model used to communicate these data in the form of standardized EHR extracts. It is necessary to remark, that there exits a one to many relationship between archetypes and integration archetypes. Given an archetype, there may be different mappings, one for each different setting that wishes to use the archetype to describe and share its data. LinkEHR is a visual tool for defining integration archetypes.

Although LinkEHR is oriented to the construction of integration archetypes it may operate as a pure archetype editor. It can load ADL files and generate both ADL and XML according to the XML schema defined by the Consortium OpenEHR. At its core lies the Archetype Object Model an object oriented model for archetypes, this has also been adopted by CEN/TC251 EN13606.

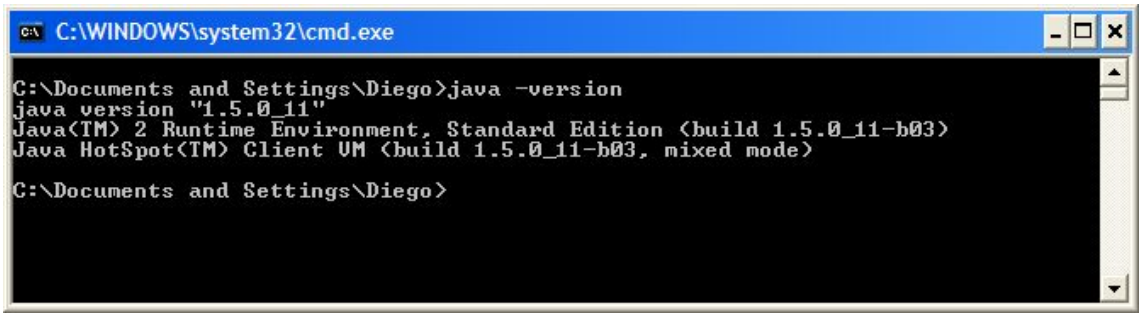
With LinkEHR new archetypes can be defined from scratch, for instance to describe the data structure and semantics of legacy data such as messages or database schemas. It is also possible to define new archetypes by specializing or altering existing ones, such as those drawn from a public available archetype repository. In any case, LinkEHR is intended to support the mapping to data sources.

NOTE: *LinkEHR has two main modules:*

1. *Multireference model editor of archetypes.*
2. *Integration archetype editor which allows the mapping of archetypes with legacy EHR data and the generation of XQuery standardization programs.*

2. Installing LinkEHR

LinkEHR is built on top of the Eclipse platform, and thus is based on Java and needs a Java virtual machine installed on the system. The first thing to do is to check if the Java VM is installed on the system. We can do that by simply typing "*java -version*" on a console (On windows is possible to open a command window by pressing *Start -> execute…* and there type *cmd.exe*)


C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\Diego>java -version
java version "1.5.0_11"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_11-b03)
Java HotSpot(TM) Client VM (build 1.5.0_11-b03, mixed mode)
C:\Documents and Settings\Diego>

If the VM is in fact installed then something like the last figure should be shown. LinkEHR works with a VM version of 1.5 or later. If a VM version below 1.5 is installed or no VM is installed at all then we must install a new version of the Java virtual machine.

To do that we must access the official java webpage and download the VM which corresponds with our OS. (http://www.java.com/en/download/manual.jsp).

To buy and download LinkEHR just visit http://www.linkehr.com.

Once the file is downloaded we must extract it. LinkEHR comes in a zip, so a tool like Winzip (http://www.winzip.com), Winrar (http://www.rarlab.com/download.htm) or 7zip (http://www.7-zip.org/) is needed to extract it to a folder. Once extracted LinkEHR can be executed by double clicking "LinkEHR.exe".

3. Archetype Edition: step by step tutorial

This guide will show up the basis of LinkEHR operation by showing you how to import both OpenEHR and CEN EN13606 reference models into LinkEHR. Later we will introduce how to open an archetype, a creation of a new archetype and the specialization of an archetype.

4.1 Running LinkEHR for the first time

The first time you open LinkEHR you will see this window

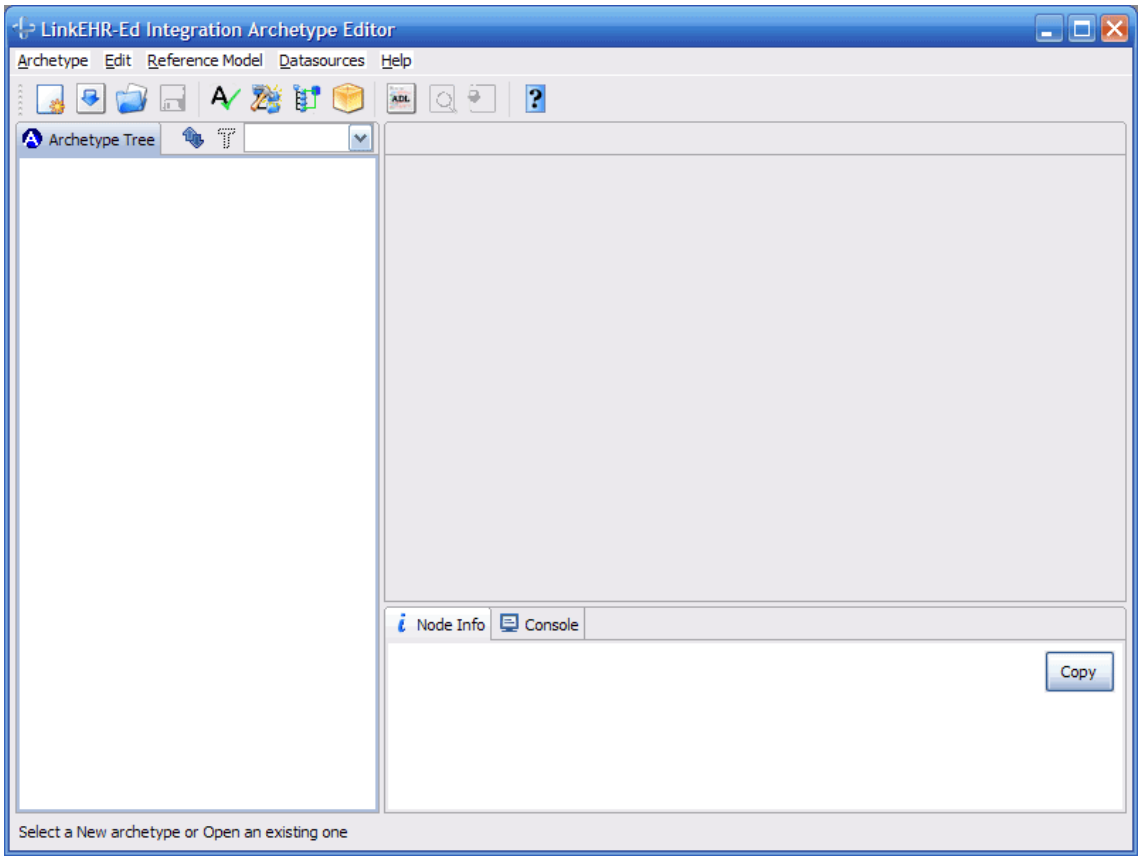


Figure 1: LinkEHR first run

LinkEHR toolbar has available the most common used actions. Figure 2 shows the meaning of buttons, although they should be auto explicative enough.

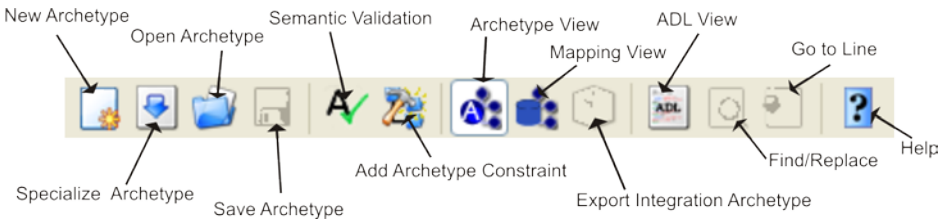


Figure 2: Buttons of the toolbar

4.2 Configure LinkEHR

User preferences can be changed in LinkEHR's configuration dialog. This dialog is located at *Help* → *Configure LinkEHR*. The dialog has four tabs, each one controlling a part of LinkEHR (see Figure 3).

a. **Languages tab**: controls the interface language of LinkEHR. Currently LinkEHR is only available in English and Spanish, but we plan to release it on several languages more.

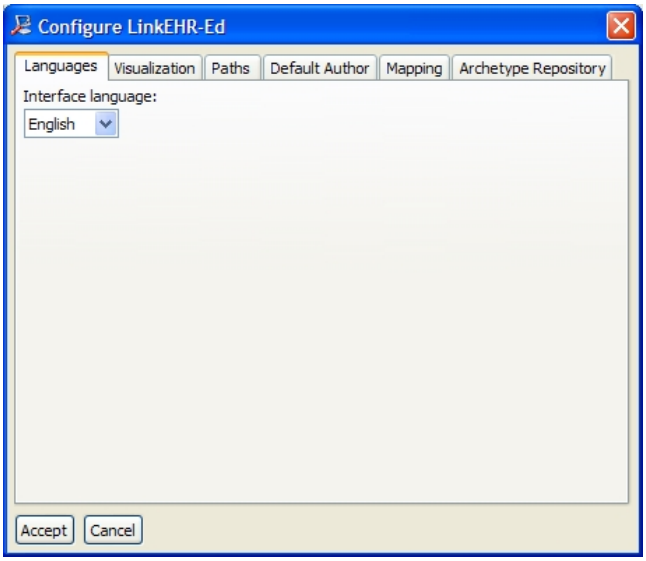


Figure 3: Interface language configuration tab

b. **Visualization tab**: you can define the preferred visualization of the archetype tree (in order to show the ontology description, classes' names or both) and the default font for the archetype tree. This also can be configured by pushing the button in the archetype tree view at any time during the archetype edition.

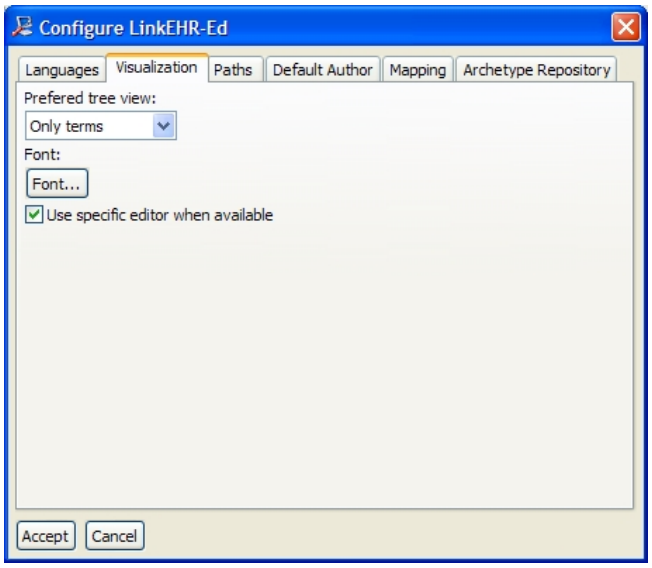


Figure 4: Visualization tab

c. **Paths tab**: you can change the paths used by LinkEHR, such as the internal (physical) repository path and the location of language and terminology XML files.

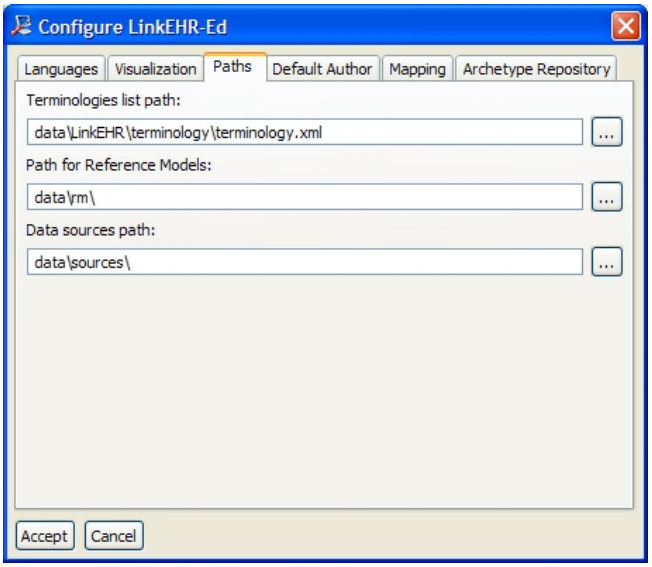


Figure 5: Path configuration tab

d. **Default author tab**: You can insert the default information of the author in order to avoid retyping it every time you create a new archetype. The Default language refers to the primary language selected by default when creating a new archetype through the “Create new archetype” wizard.

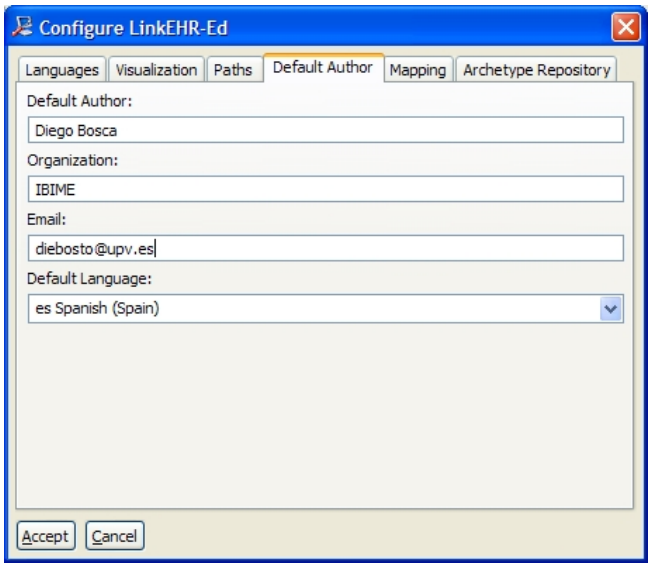


Figure 6: Default author information tab

e. **Mapping tab**: Some options for mapping edition

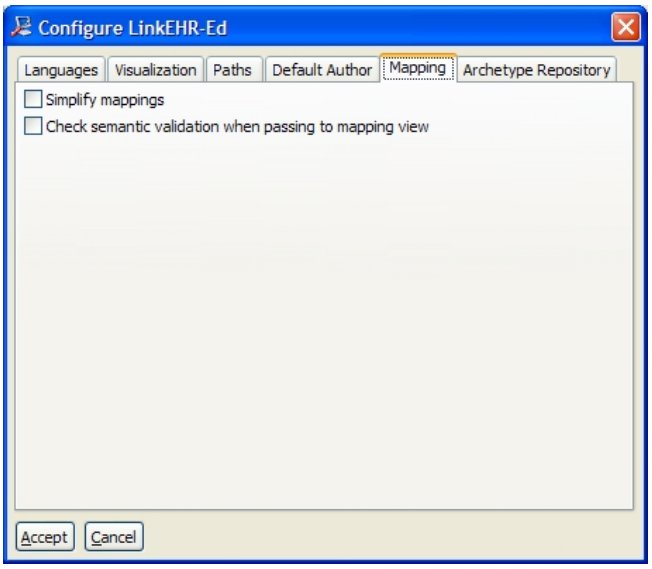


Figure 7: Mapping options tab

- Simplify mappings: Changes the mapping to its simplest form.
- Check semantic validation when passing to mapping view: Tells LinkEHR if semantic validation should be performed while passing to mapping view.

f. **Archetype repository**: Options to configure the archetype repository where archetypes are stored.

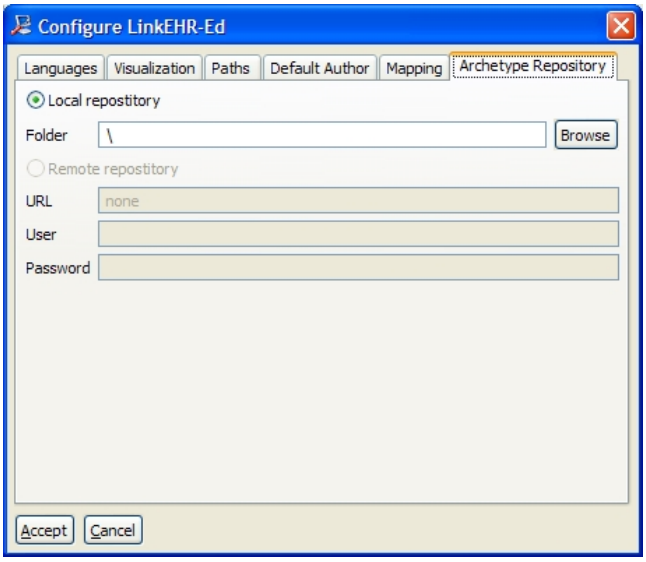


Figure 8: Repository configuration tab

COM

4.3 Adding a new reference model to LinkEHR

To import a new reference model you have to open the import reference model wizard, which is located on the under *Reference Model → Reference Model Manager → Import RM → From XSD Schemas.*

> ***NOTE**: LinkEHR has also the option of importing the reference model from a special compressed file which contains all needed files. For this example we will import the new reference model from their schemas.*

In the reference model import wizard you can introduce the organization name, the model name and the XSD schemas defining the reference model, as you can see in Figure Figure 9.

Since we are constraining an OO model, the distinction between classes and attributes is crucial. In fact, different types of constraints are applied accordingly. For instance, classes may be represented as elements or as type definitions. In the former approach, XML instances contain elements tagged with the class names while in the latter class names only appear in the schema. When importing a new reference model, it must be indicated whether or not classes are represented as elements in the schema. It is supposes that all classes are represented in the same way.

In the case of the OpenEHR official XML Schemas, you must set the combo “Represent complex elements as” to Complex types.

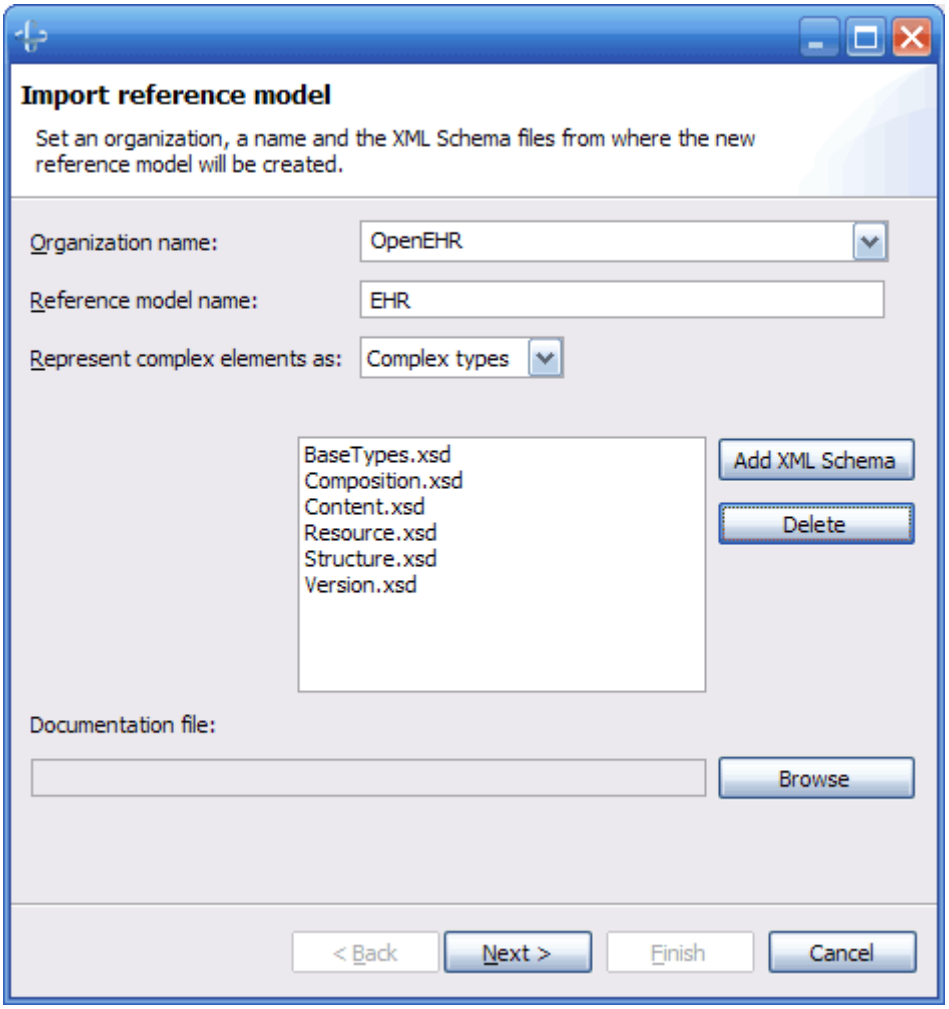


Figure 9: Importing OpenEHR reference model

Once all this information has been introduced you can push Next button. In the next wizard page will appear the list of entities available to be generated. The list is filled with the entities in complexity order, as it is likely that the more complex entities will be the ones basic to that model, also called Business Objects. If a needed entity does not appear in the list you can move the slider in order to reveal less complex entities. They are hidden at first to simplify the whole process.

When you have all the desired entities visible you only have to select them and click on the arrow to the right. This will show the selected entities on the “included” list.

***NOTE**: Included entities will be the only ones available as basis for creating new archetypes at the* New archetype *dialog.*

When at least one entity is included you can finish the importation process. Figure Figure 10 shows the entities of OpenEHR reference model.

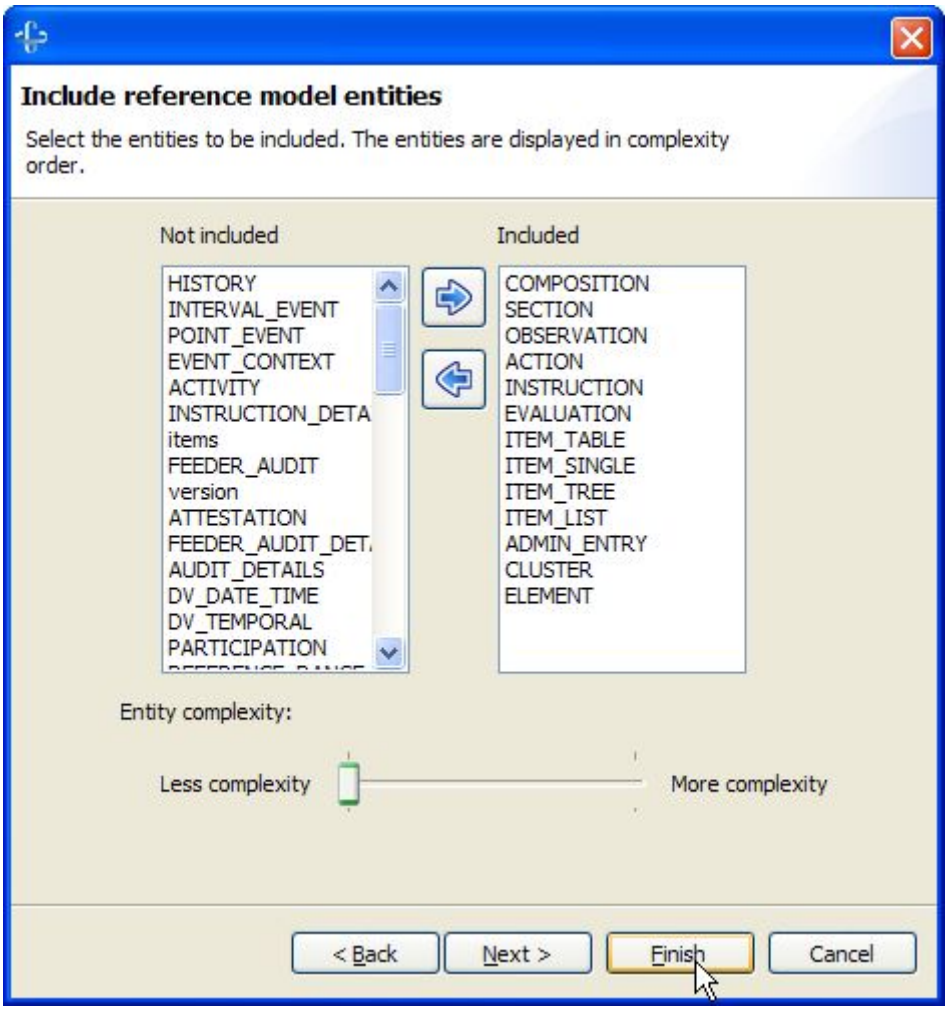


Figure 10: The entities of OpenEHR reference model

Now we can check that the imported reference model has been included by trying to create a new archetype. Push *Archetype* → *New* and the window of Figure 11 will appear. See section 5 for more details.

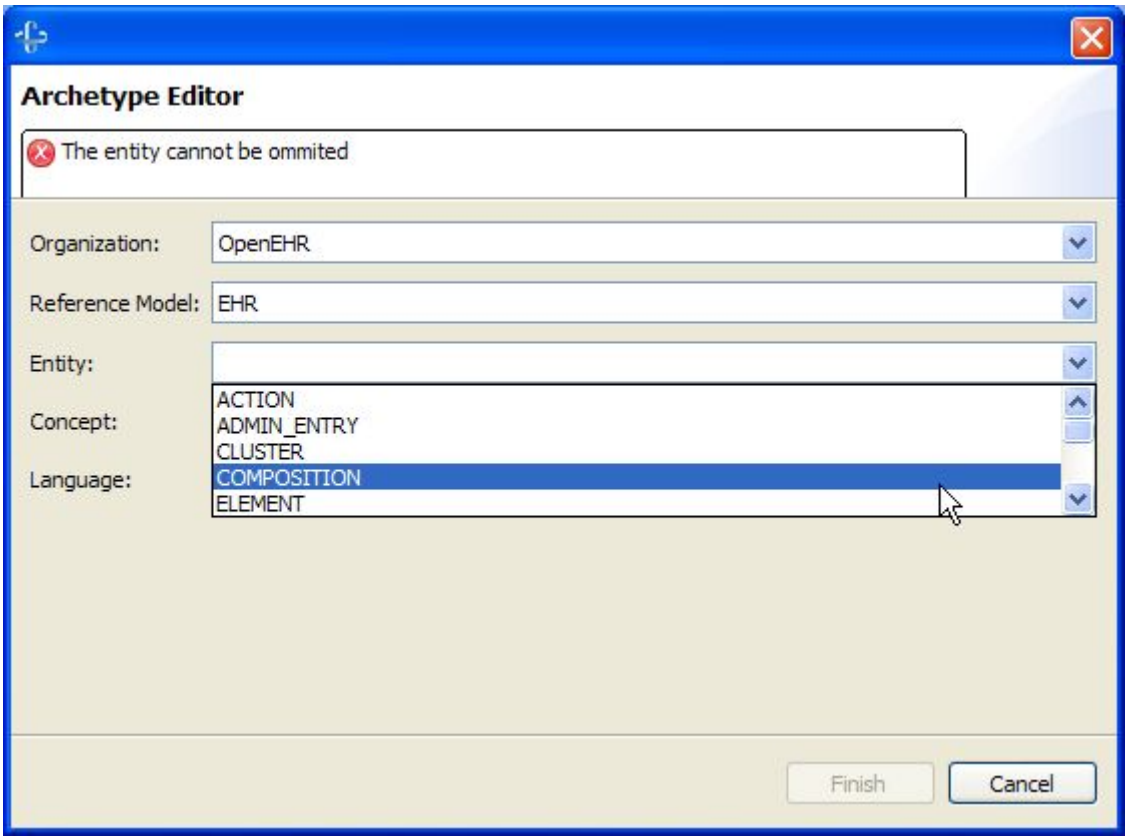


Figure 11: The new archetype wizard with the OpenEHR reference model included.

In the same way we have added the OpenEHR reference model we can add the CEN EN13606 reference model to LinkEHR. So the first page will be as shown in the Figure 12.

> ***NOTE**: There is no official XML Schema for CEN EN13606. We have developed our own schema and it is available through the LinkEHR project webpage.*

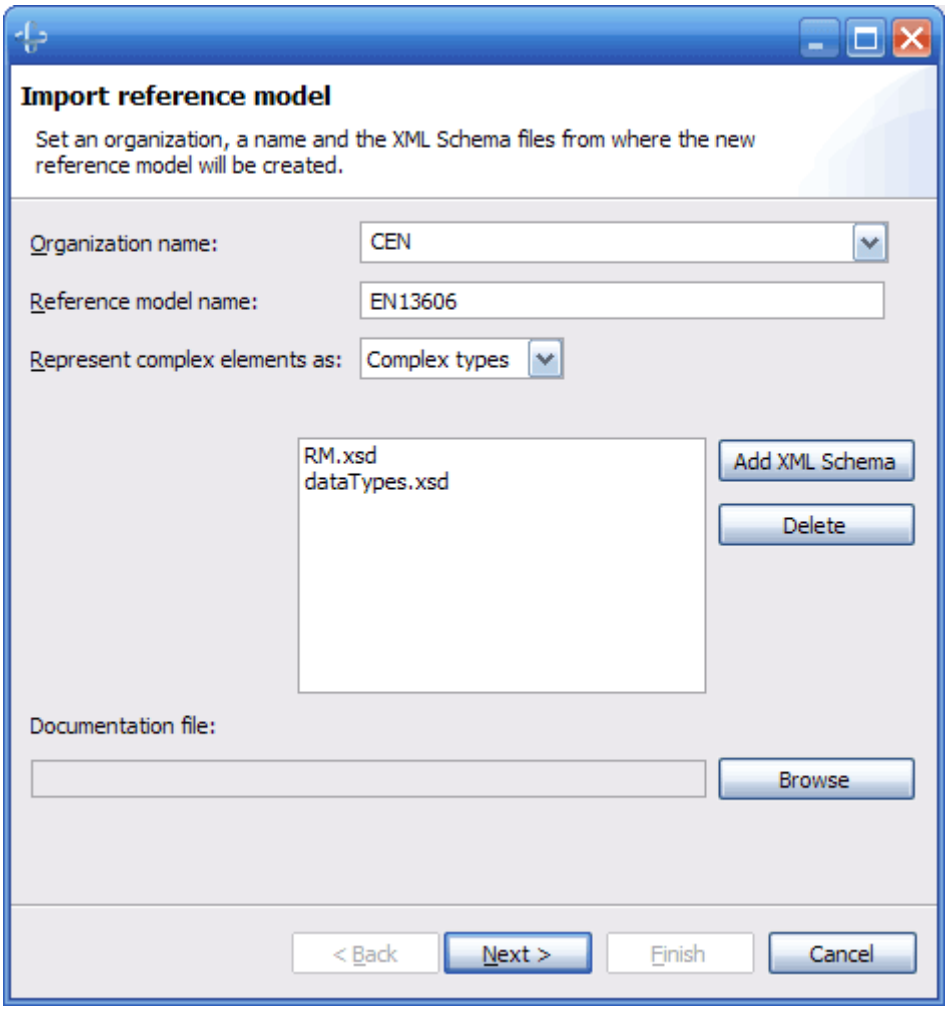


Figure 12: Importing CEN EN 13606

In the same way as before, we select the reference model entities to be included and push finish. The entities needed for EN13606 are shown at Figure 13.

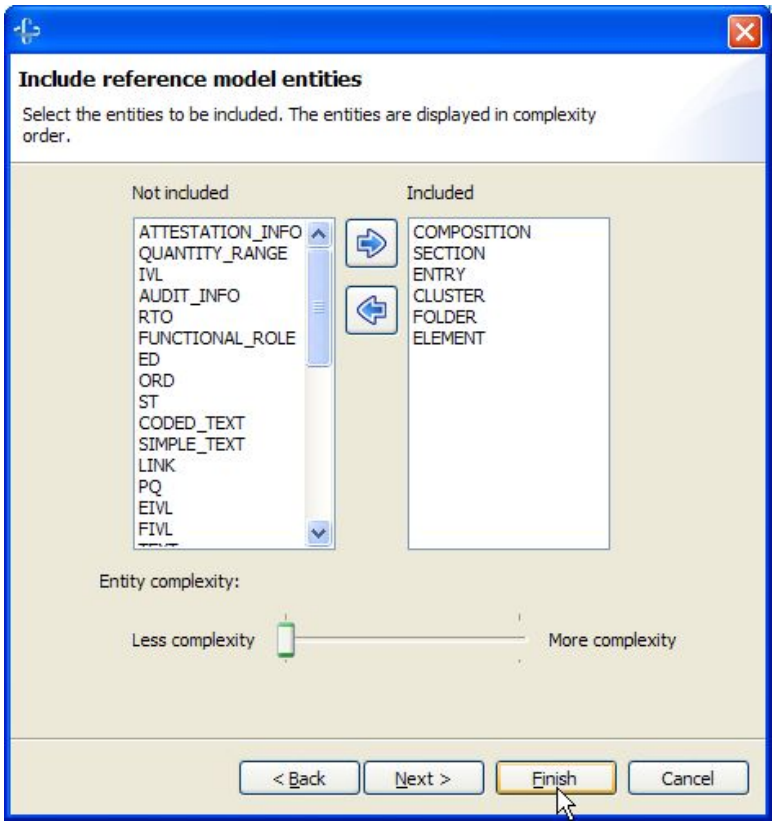


Figure 13: Reference model entities for CEN EN13606

4.4 Creating a new archetype

To create a new archetype we only have to push the New Archetype button of the toolbar or the menu *Archetype* → *New* and a wizard (shown at Figure 11) will be opened. After selecting the organization, reference model, entity, concept and language a new and empty archetype will be loaded and ready for being edited (Figure 14).

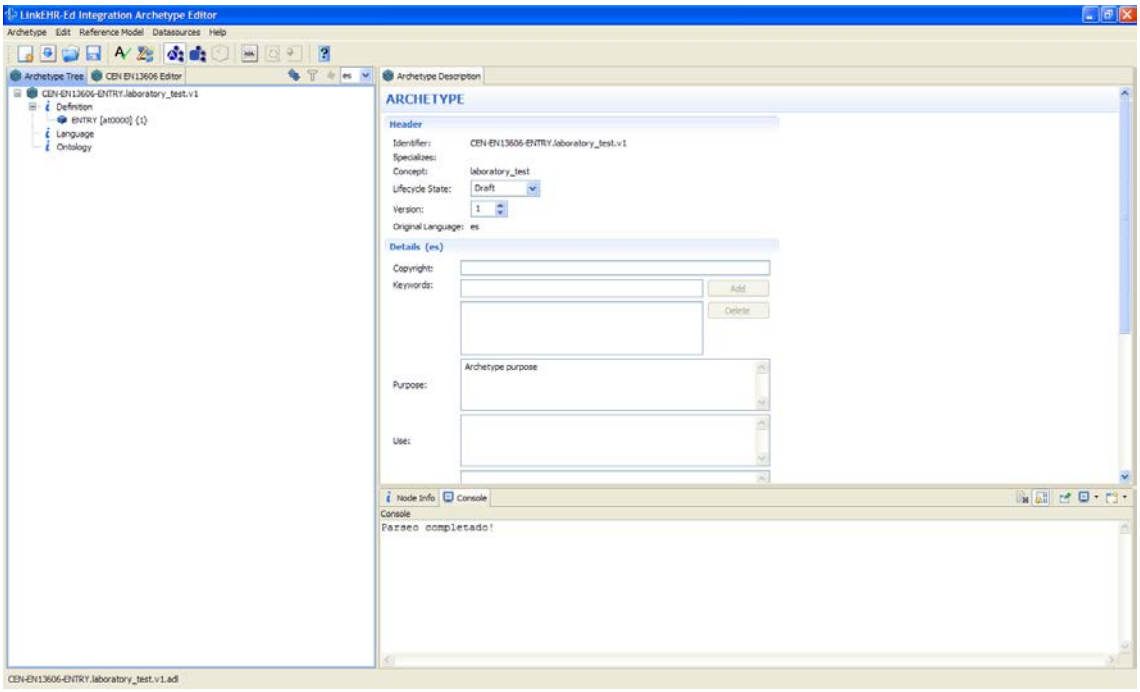


Figure 14: LinkEHR with a new archetype created.

4.5 Opening an archetype

This process is very straightforward. By pushing the Open button on the toolbar or the menu *Archetype* → *Open* a typical system open dialog will be shown. Choosing then an ADL file (version 1.4) will load it into LinkEHR.

If the ADL is not valid, an error will be shown indicating the line where the error has been detected. Moreover, although the archetype tree can not be built the ADL code is available and can be accessed through *Go to ADL* button (represented with this icon).
Looking at the ADL code will allow you to easily detect the error, fix it and recompile the ADL text (by pushing the button again) to generate the visual representation of the archetype.

4.6 Editing an archetype

The main window of LinkEHR has four main components or views:

a. **Archetype tree view**: Here is where the archetype structure is represented as a tree, including its header and description, the definition tree, the language section and the ontology section (Figure 15). The button switches the definition branch of the tree between technical and non-technical view.

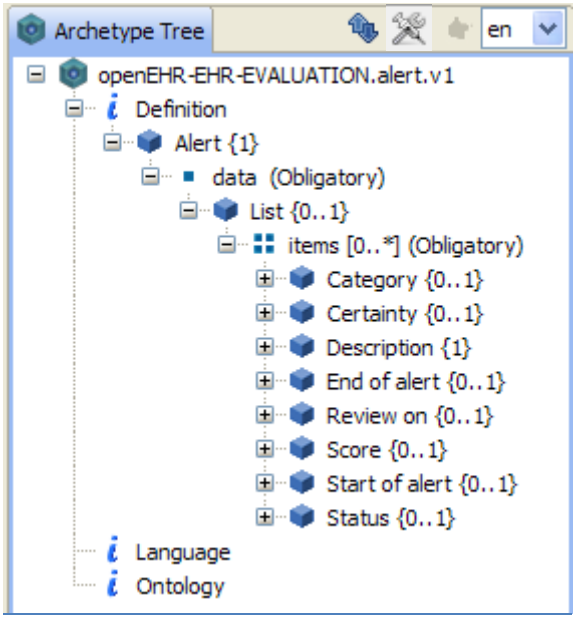


Figure 15: The archetype tree containing all of its structure

b. **Details view**: At the right side of the screen is where the different forms for introduction of data are loaded. A different form is associated to each kind of node of the archetype tree (see Figures 16, 17 and 18).

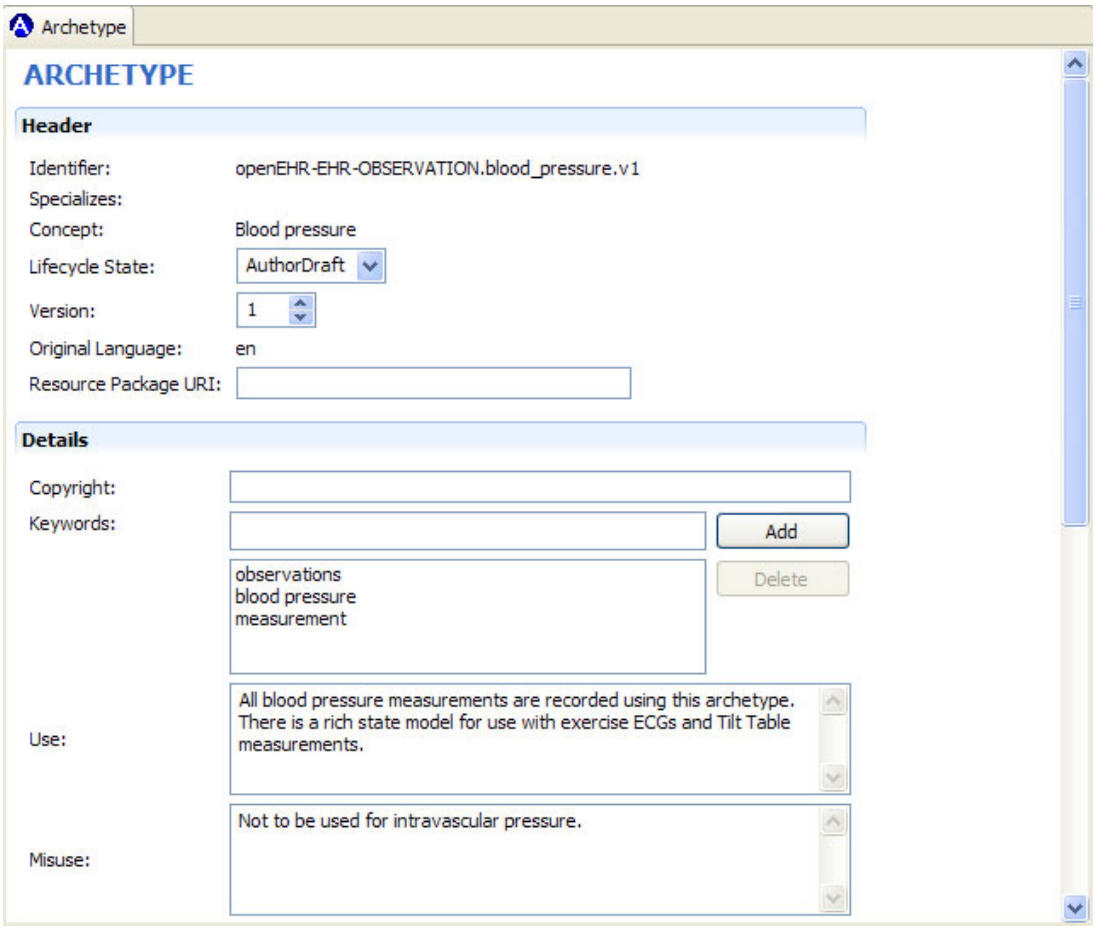


Figure 16: The archetype header a description form

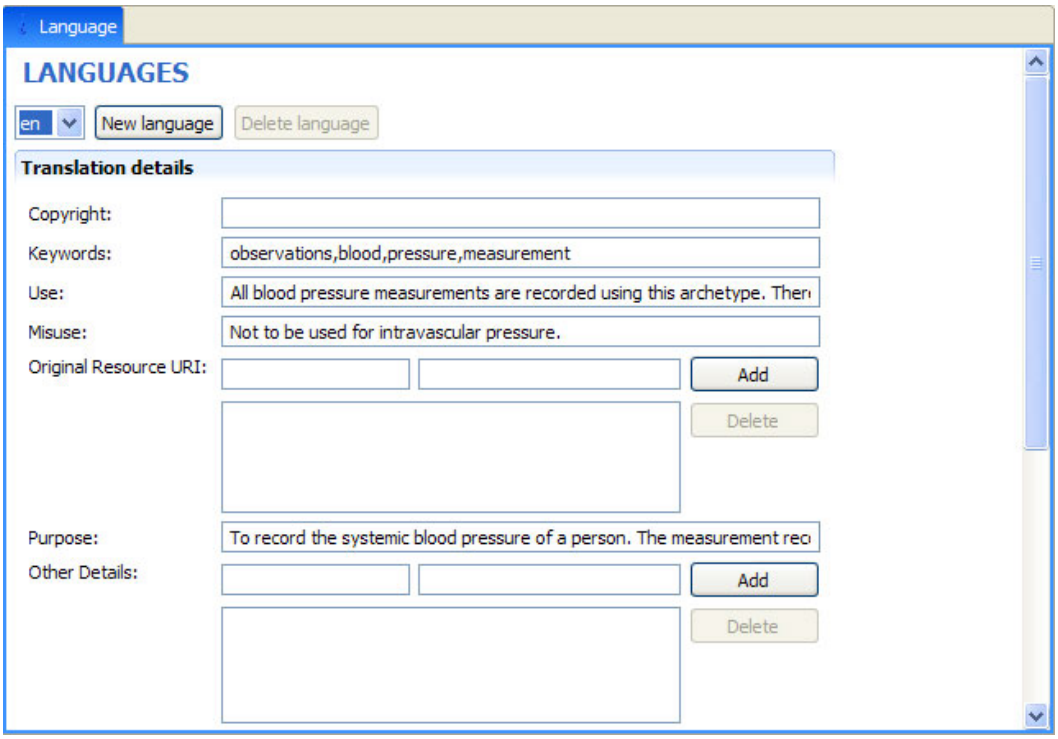


Figure 17: The language section form

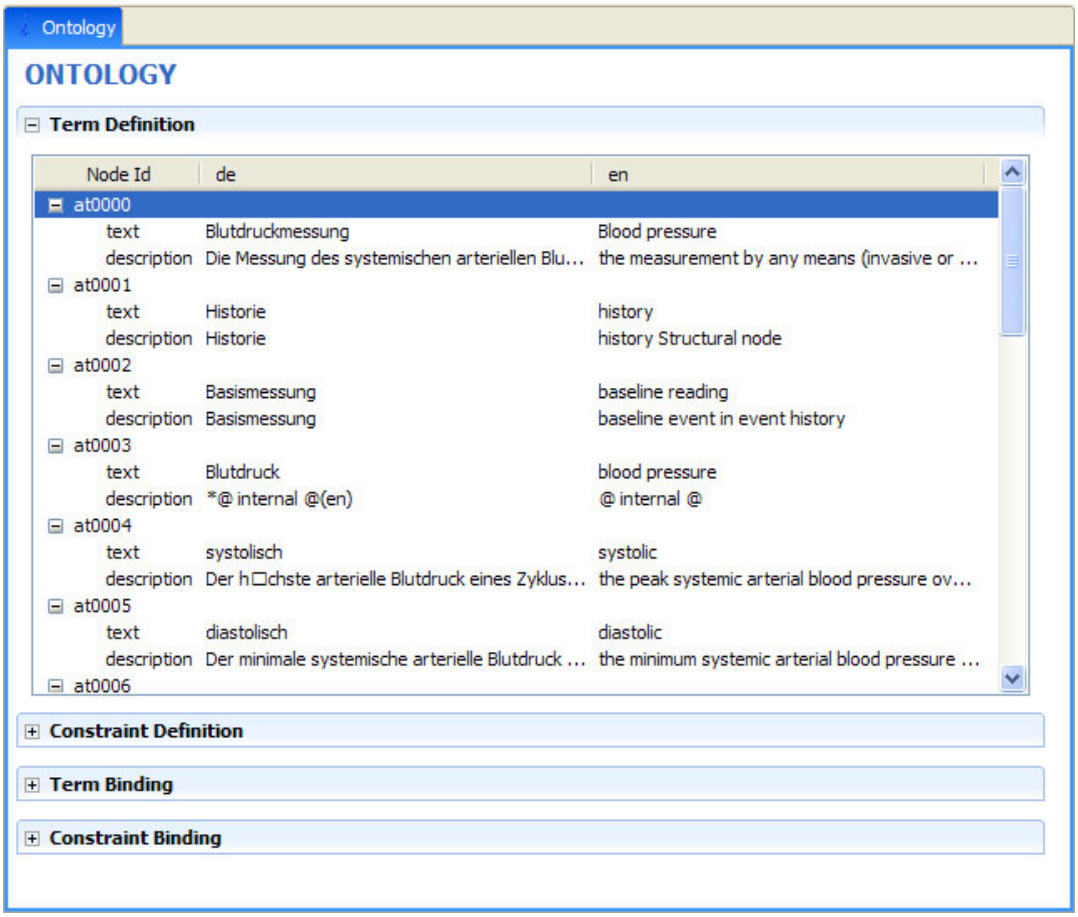


Figure 18: The ontology section form

c. **Console view**: This view shows the application messages and also has a tab to watch the path of the selected node of the definition tree.

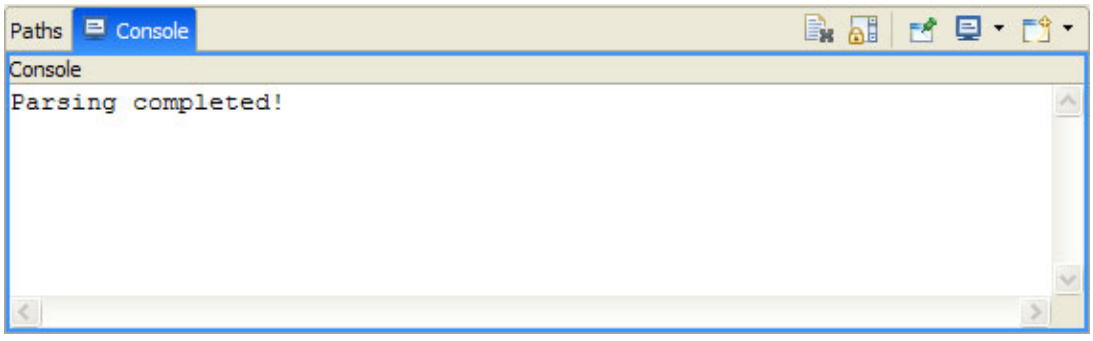


Figure 19: The console view

d. **ADL view**: Once an archetype has been loaded or created, you can always switch to the ADL view in order to see and edit it. Every change made in the ADL will be reparsed and validated before going back to the visual

representation. You can switch to the ADL view by pushing the button,

and return to the previous view by pushing it again.

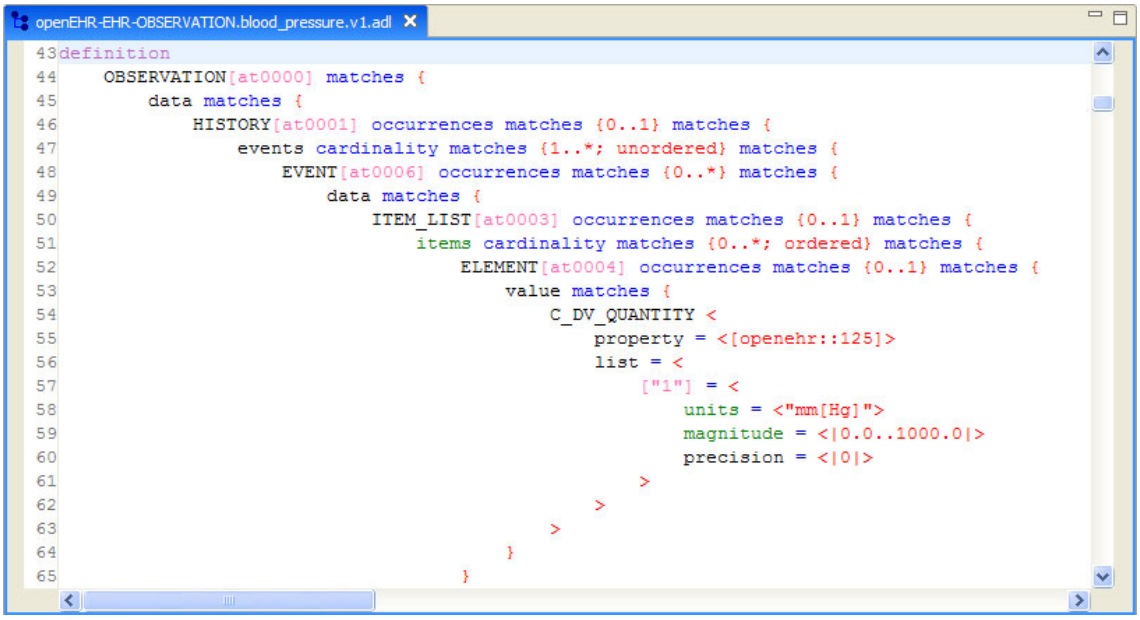
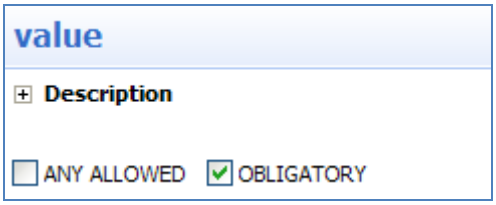


Figure 20: The ADL edition view

The main edition process is done at the definition branch of the archetype tree. By clicking on any node a form is shown in the details view with the editable information of the node. This form will depend on the type of node:

- **Attribute**: There are views for editing properties of Single and Multiple attributes.



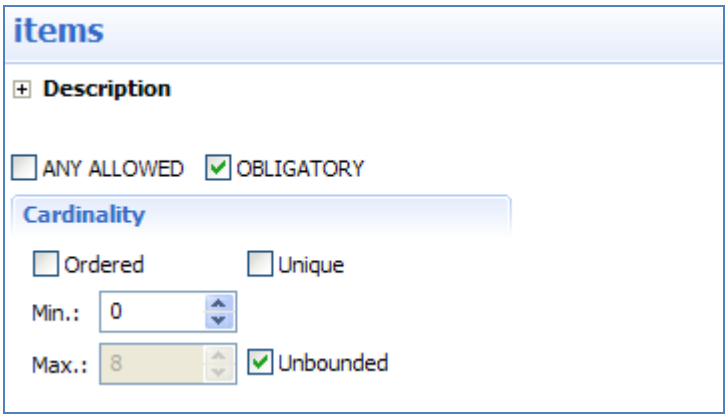


Figure 21: Single and multiple attributes view

- **Complex Object**: Occurrences of complex objects can be modified through their respective form. Their ontology information (term definition and term binding) can also be edited from this form.

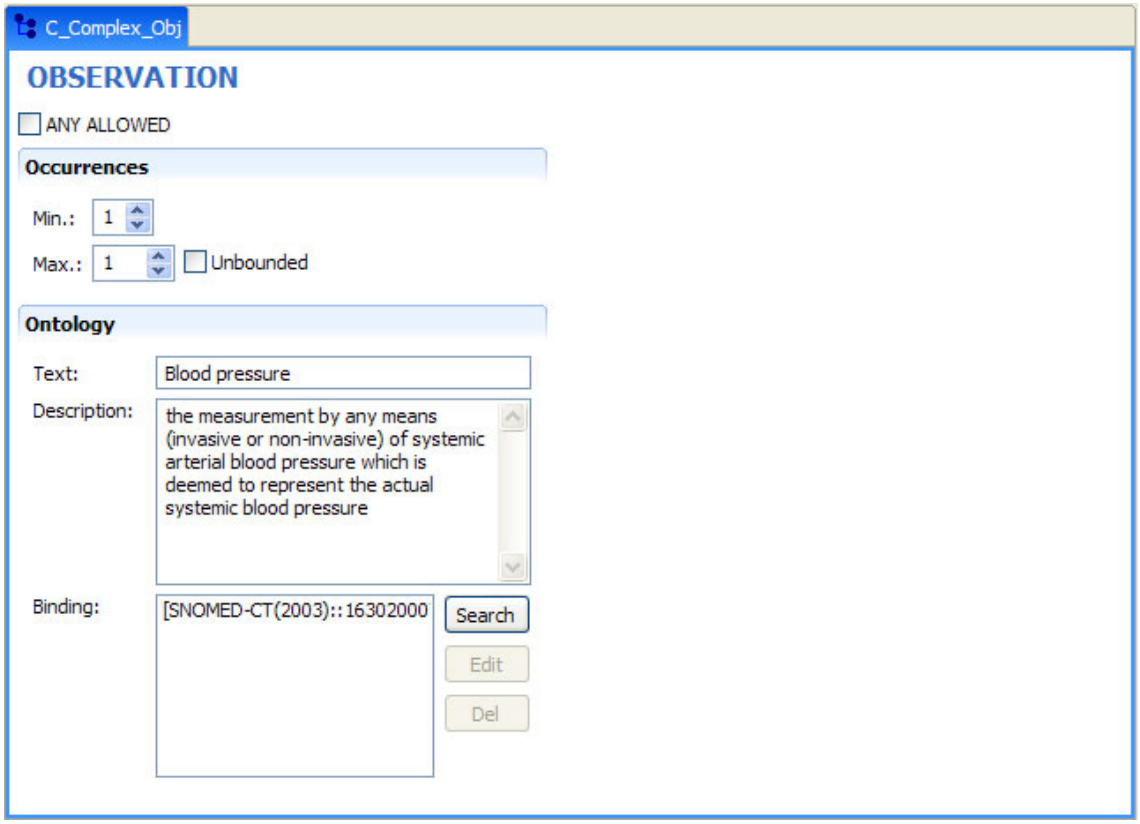


Figure 22: CComplexObject edition view

- **Internal Reference**: When an internal reference is created, a list of the available target nodes of the same reference model type is provided. You can change the target node at any time or directly jump to its definition.

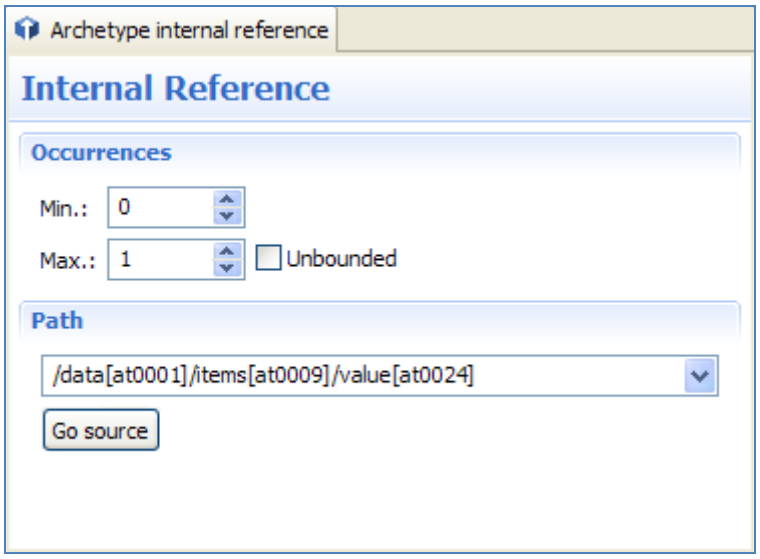


Figure 23: Internal reference view

- **Archetype Slot**: Properties of an archetype slot (includes and excludes) can be edited from this form.

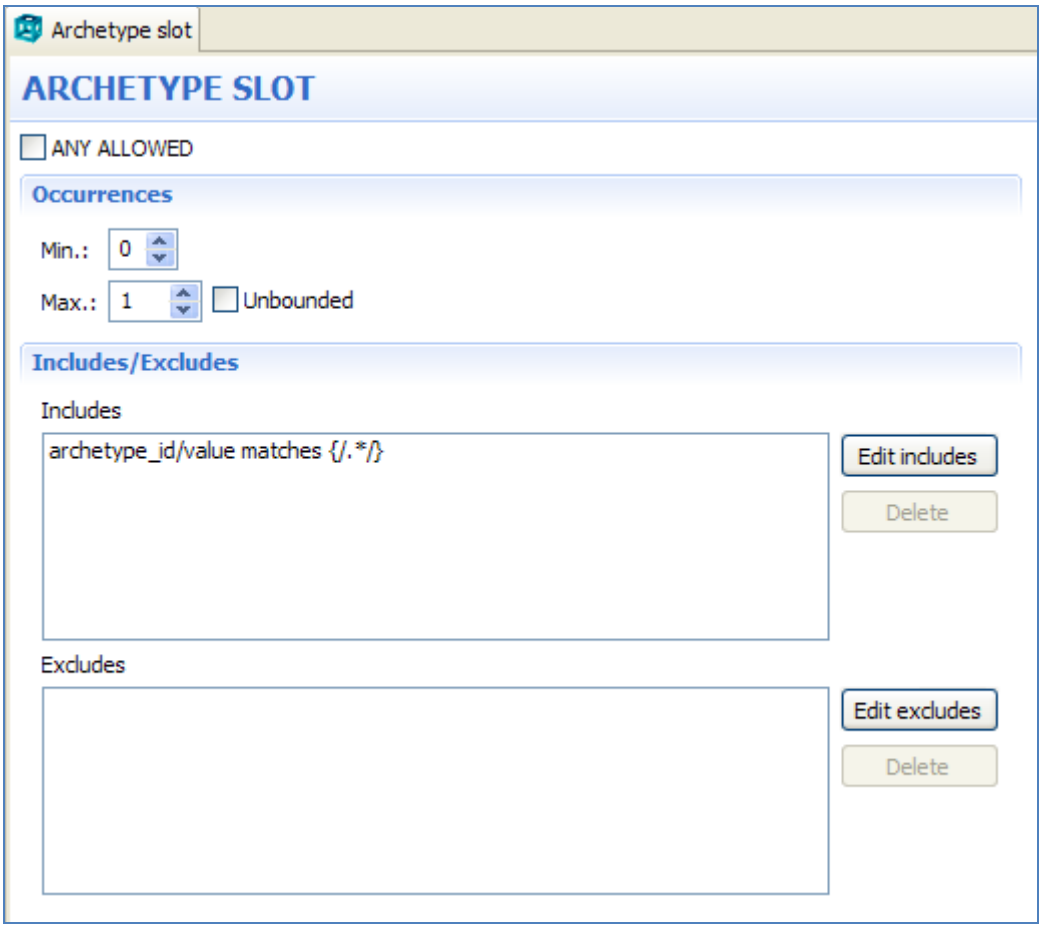


Figure 24: Archetype slot view

- **Primitive Object**: Each kind of primitive objects (corresponding to the basic types string, integer, double, boolean, date, time and datetime) has its own form in order to define their constraints or assumed values.

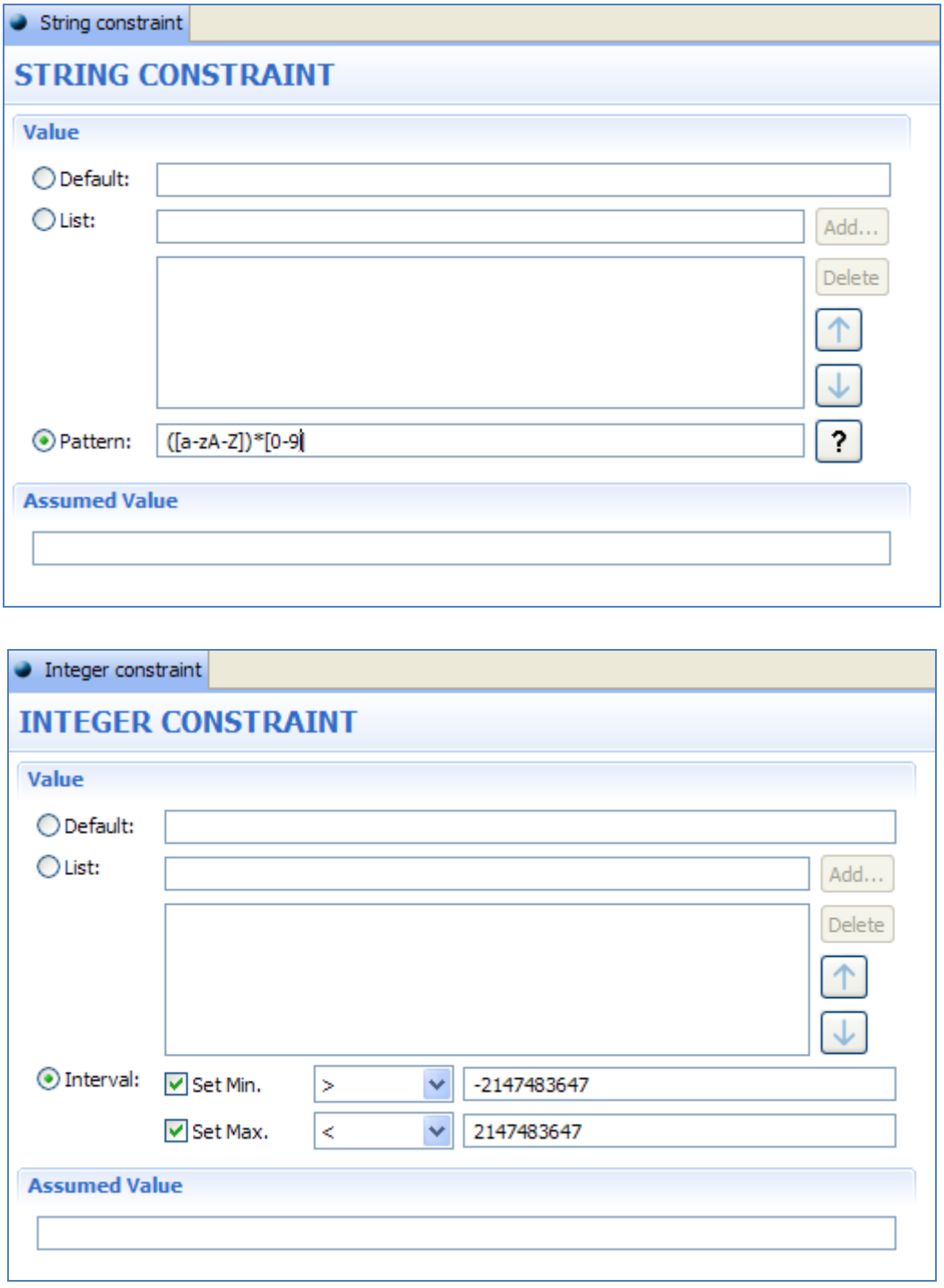


Figure 25: Samples of String and Integer primitive objects views

- **Domain Types**: Since LinkEHR is a model-independent editor, it is difficult to design an interface for editing particular domain types. Due to this fact, at this stage there is no visual interface for this kind of objects, but they can still be edited textually by switching to the ADL view. The interfaces for the domain types could be defined on specific model editors (see section 4.11 and 4.12)

In order to edit the archetype you can add or delete new objects and attributes by clicking on the "Add archetype constraint" button of the toolbar ( ) or by right-clicking on the node of the tree where do you want to perform those actions (Figures 26 and 27). In the menu that shows up you can choose create an object, an internal reference to an object, or an archetype slot.

You can delete a node (objects and attributes) selecting the *Delete* option from the right-click menu over the corresponding node.

***NOTE**: When two or more objects are assigned to a Single Attribute, a virtual node called Alternatives will be created in the tree in order to show this specific fact. It is only an informative node and has no other properties.*

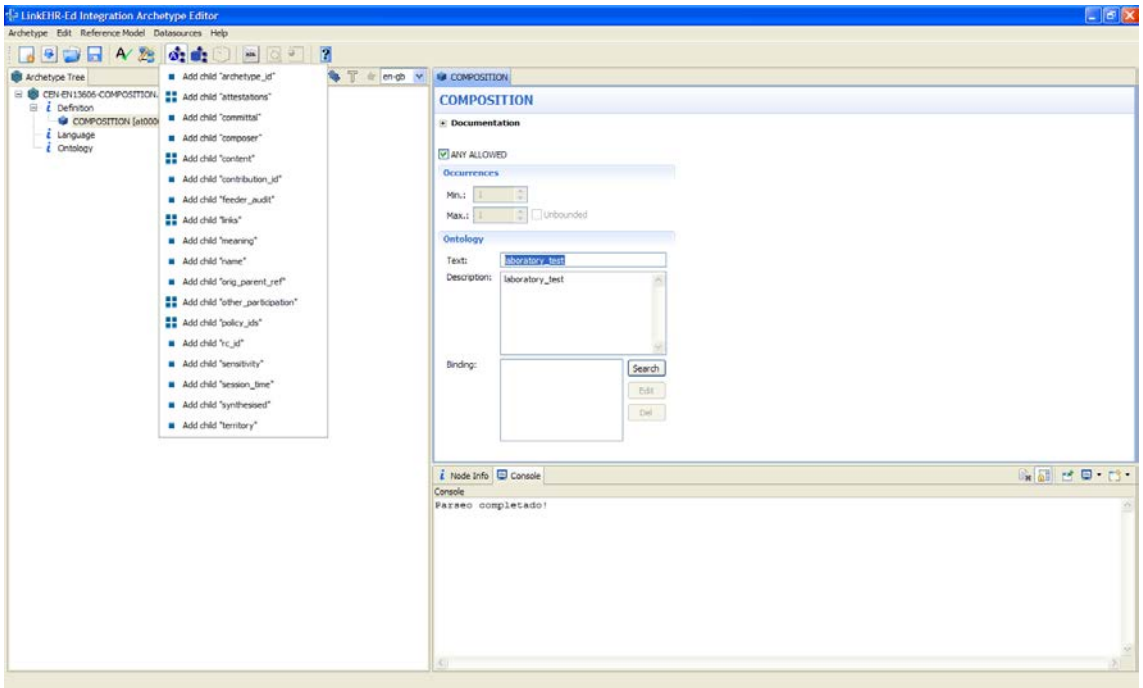


Figure 26: Pushing the "Add archetype constraint" button shows a menu showing the available actions…

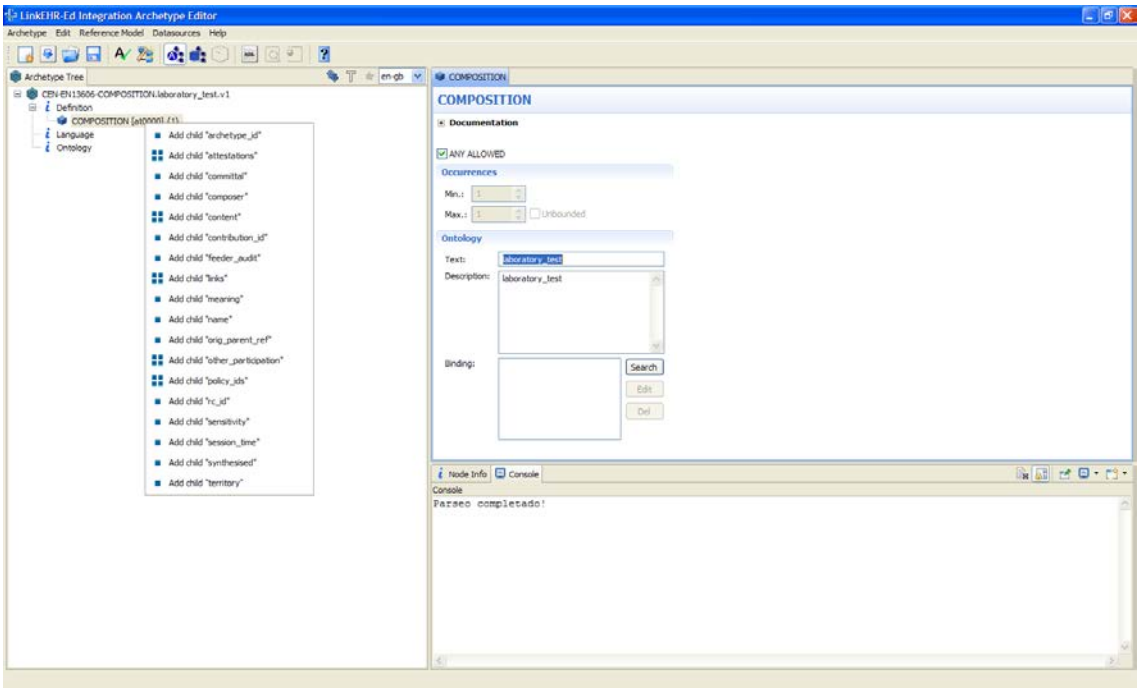


Figure 27: …and so does clicking with right button on a node

COM

4.7 Specializing an archetype

To create a specialization archetype you only have to push the "Specialize Archetype" button of the toolbar or the menu *Archetype* → *Specialize*. A dialog will be displayed to select the parent archetype of the specialization and to introduce the specialization name. Pushing Ok will create the specialization archetype, with the structure of the parent archetype already included. Then you can modify the specialized archetype the same way as told at step 7.

COM

4.8 Validating an archetype

Once you have created an archetype from scratch or opened it from an available ADL file, you can validate it by pushing the Validate button . This validation assures that the introduced constraints are effectively more constrained than those defined by a parent archetype (if this is a specialized archetype) or by the reference model. In any other case, an error message will be displayed with the path of the node where the validation failed.

4.9 Save an archetype

Once you have finished the edition of an archetype, you can save it trough the Save button or the *Archetype* → *Save* menu. You can choose an ADL format or an XML one (just a prototype functionality in the second case).

COM

4.10 Reference model specific editors

LinkEHR has the feature of being able to use specific editors for each one of the available reference models (in addition to the generic archetype editor). At this moment, a CEN EN13606 specific editor (See point 6 for more information about CEN EN13606 editor) has been defined in order to ease the archetype definition for this norm. New specific editors can be defined for any reference model (See point 4.12).

4.11 Defining a reference model documentation

LinkEHR can also help in the generation of reference model documentation. This can be done in the Reference Model Manager (which can be accessed from the menu *Reference Model* → *Reference Model Manager*). Figure 28 shows the Reference Model Manager dialog.

Reference Model Documentation includes not only a short description of the classes and attributes of the reference model, but also provides an easy way to define specific editors for a desired reference model (See point 11).

This dialog has three different parts. First part shows the available reference models already imported to LinkEHR (For a step by step guide to reference model importing, please check points 3 and 4 of this manual). The second part contains the list of objects and attributes already defined for the selected reference model. Last part contains a form where the values for the objects and attributes can be changed. This part has also the properties for the current reference model.

- *Model Class* is the class name and path that will be used to generate objects on the specific editor
- *XSLT Path* is the path to the XSLT file to transform the result XML schemas for this reference model

To generate the documentation to the different nodes first you have to create the objects and its attributes by pressing the *Add Node* button. As the objects and attributes have been created, you can then change their properties. Both classes and attributes have a *Short description* and a *Long description*. *Short description* is intended to be used as an instant help (e.g. tooltips) while *Long description* is a more exhaustive definition of the node.

For the classes, the following properties can be changed:

- *Is son of:* Choose which class is the parent of current class.
- *Is abstract:* Choose if current class is abstract
- *Form class:* Choose the class that contains the form to edit this kind of class (this class will be launched by LinkEHR when editing this class)

- *Icon path:* The path to the icon of this kind of class
- *Navigation attribute:* The name of the attribute used to navigate to other main classes of the reference model (e.g. on a COMPOSITION class, *Navigation attribute* is “content”)
- *Allowed types:* Types that this generic class can handle (e.g. The *Allowed types* for IVL<T> are TS, PQ, DURATION…)

For attributes, following properties can be changed:

- *Terminology:* Choose the terminology for this attribute
- *Visible:* Choose if an attribute is on the way from a reference model class to a leaf class (typically a data type).
- *Allowed children:* List of all classes that can hang from this attribute

***NOTE**: Keep in mind that specific editors should be kept simple enough to ease the archetype creation for non-technical people.*

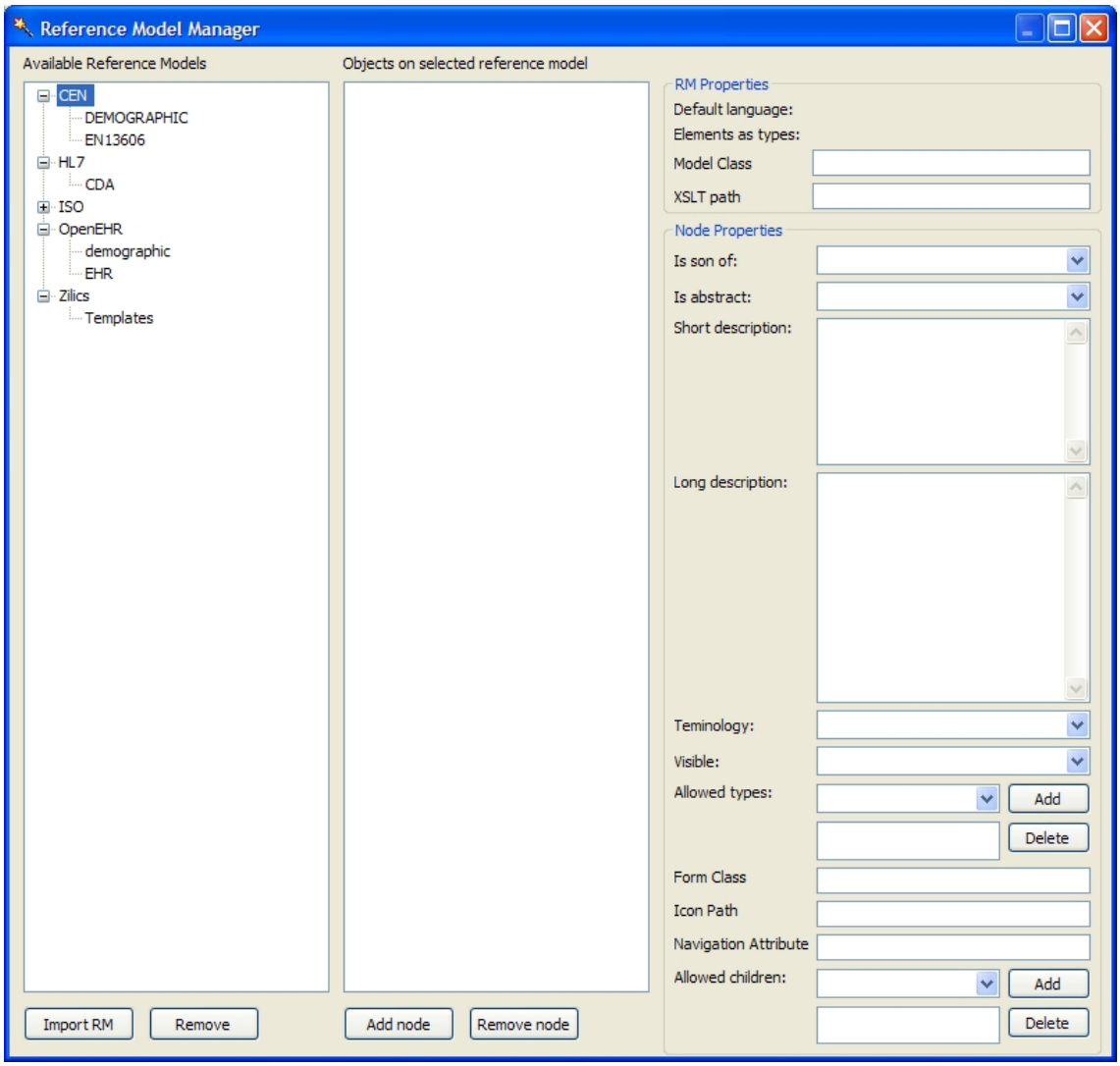


Figure 28: Reference model manager

4.12 Translating archetypes to another language

As was shown in Figure 17, it is possible to define new languages for the archetype anytime. This will allow translating all objects texts and descriptions just changing the current language combo from the archetype tree (see Figure 29). Text and descriptions can also be changed directly from the ontology view (Figure 18).

***NOTE**: Archetype details section is also language dependant, so changing language on the archetype tree will allow changing the details for that language.*

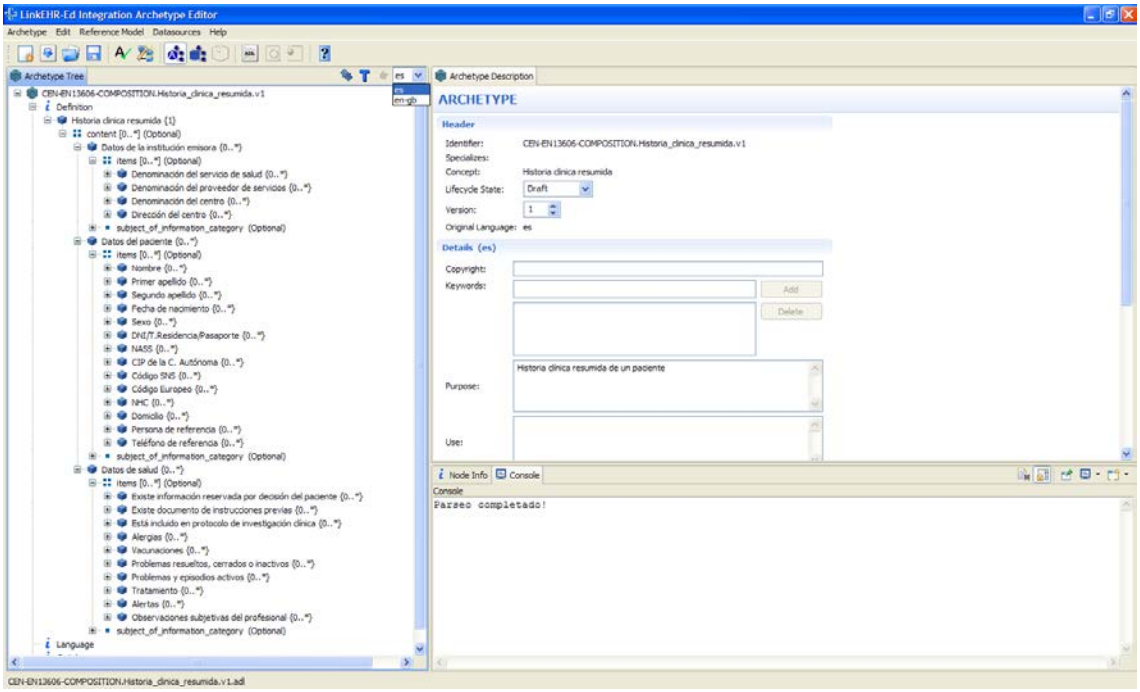


Figure 29: Changing between multiple languages of an archetype

COM

5. Mapping archetypes to data sources

Archetypes are formal definitions of domain concepts based on a particular reference or data model. When we are working with legacy and non-standardized data, archetypes can be considered as a view that provides abstraction in interfacing between data sources and reference models used to communicate clinical data.

Since health data to be made public resides in the underlying data sources, it is necessary to define some kind of mapping information that links entities described in the archetype (object nodes and attributes) to data elements in data repositories (e.g. elements and attributes in the case of XML documents, tables and attributes in the case of relational data sources). An *integration archetype* is considered to be a mechanism that binds an archetype definition with existing data schemas. Therefore:

Integration Archetype = archetype + mapping specification

5.1 Comprehensive archetypes

Imagine a very simple reference model for vehicles which states that a vehicle can have one to many wheels and transport one to many passengers (Figure 31).

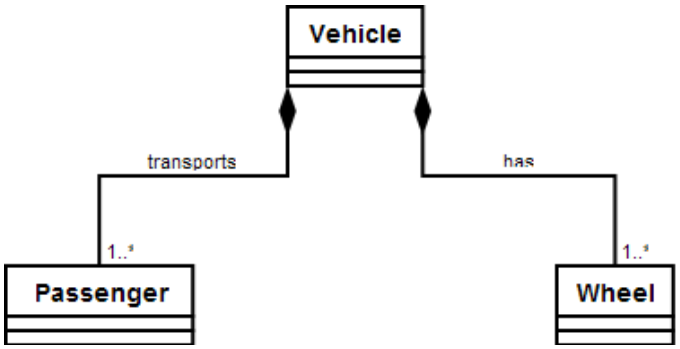


Figure 31. Vehicle reference Model

We can define a new archetype, for example for representing a bicycle, by constraining the reference model. In this case, we define the concept bicycle as the vehicle with only two wheels (Figure 32).

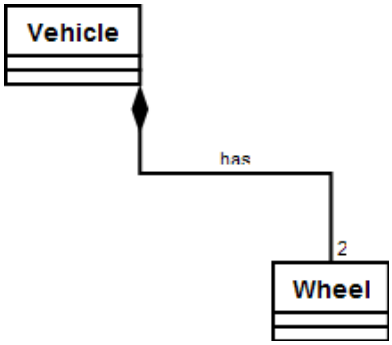


Figure 32. The archetype "Bicycle"

Finally, if what we want is to generate instances of bicycles, it is necessary to merge the constraints of the archetype with all the information that has not been constrained of the reference model (Figure 33). This is what we call *comprehensive archetype* and it is the basis or target schema that will be mapped to the original data source schemas, since we do not only want to standardize the archetype related data but also all the surrounding data, for example, the context information supported by the reference model classes.

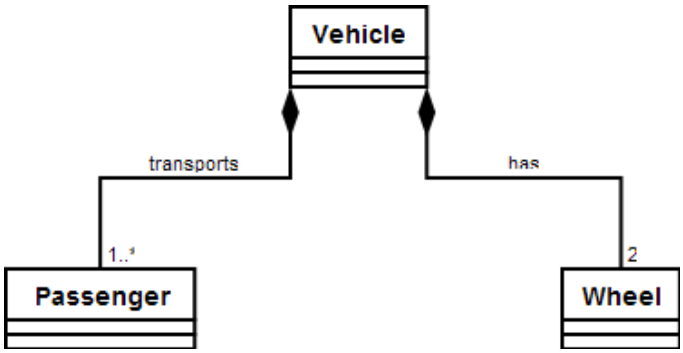


Figure 33. Comprehensive archetype: reference model + archetype

Comprehensive archetypes are automatically generated by LinkEHR when entering into the mapping view of the editor and substitute the archetype tree of the archetype edition view.

5.2 Mapping of archetypes to data sources

In our context the mapping specifications must define how to create archetype instances from the content of the data sources. It should be noted that every valid instance of an archetype is also instance of the reference model. Therefore, if the reference model is a standard, such as ISO 13606, the generated instances are compliant with that standard. Target archetypes may be created ad-hoc to mimic the legacy data or it may preexist, e.g. when it is drawn from a public archetype repository.

Archetypes are used to model arbitrary complex domain concept without any consideration regarding the potential internal architecture or database design of EHR systems. As a consequence, complex and expressive mapping specifications are necessary due to the low similarity between archetypes and EHR system. We have tried to incorporate to LinkEHR expressive mappings that not only specify value couplings but also structural mappings.

There exist two kind of mapping specifications: atomic attribute mappings and object mapping. Atomic attribute mappings define how to obtain a value for an atomic attribute of an archetype (C_PRIMITIVE_OBJECTs, i.e. attributes that contain simple data types such as strings, integers, booleans…) by using a set of values from the data sources. For this purpose transformation functions and conversion tables can be used. The simplest kind of attribute mapping is the identity function which copies a constant or a single source value into a target value.

LinkEHR requires the definition of an attribute mapping at least for each mandatory atomic attribute of the archetype. For each constrained class there exists an object mapping which contains both the query to be used to retrieve all the data necessary for generating data instances and the set of attributes that identify univocally the instances of the class. The combination of both components allows the conversion from source data to XML documents compliant with the Reference Model.

Archetype designers are responsible of defining the atomic attribute mapping and the system tries to generate automatically from them a set of candidate object mappings by taking into account the structure of the Reference Model entity, the constraints defined in the archetype and the integrity constraints of data sources. This approach alleviates the work of defining how to populate archetypes since it is easier for the designer to indicate which data elements of the data sources are relevant to a certain archetype attribute, rather than to specify the possible complex query required to extract and transform all the relevant information, which may involve many data structures possibly from several data sources.

Once all the mandatory attribute mappings are defined, LinkEHR can generate an XQuery script that will transform legacy XML data instances into standardized XML instances following the archetype definition and reference model structure.

***NOTE**: The generated XQuery script is directly applied to XML data. A mechanism to permit this application to relational data sources transparently for the user is being developed.*

5.3 Integration Archetype Edition: step by step tutorial

After opening an existing archetype or creating a new one (please, see the achetype edition manual also available at the web page), to perform the mappings we must enter into the *mapping view* by pushing the button of *Map View* (Figure 34):

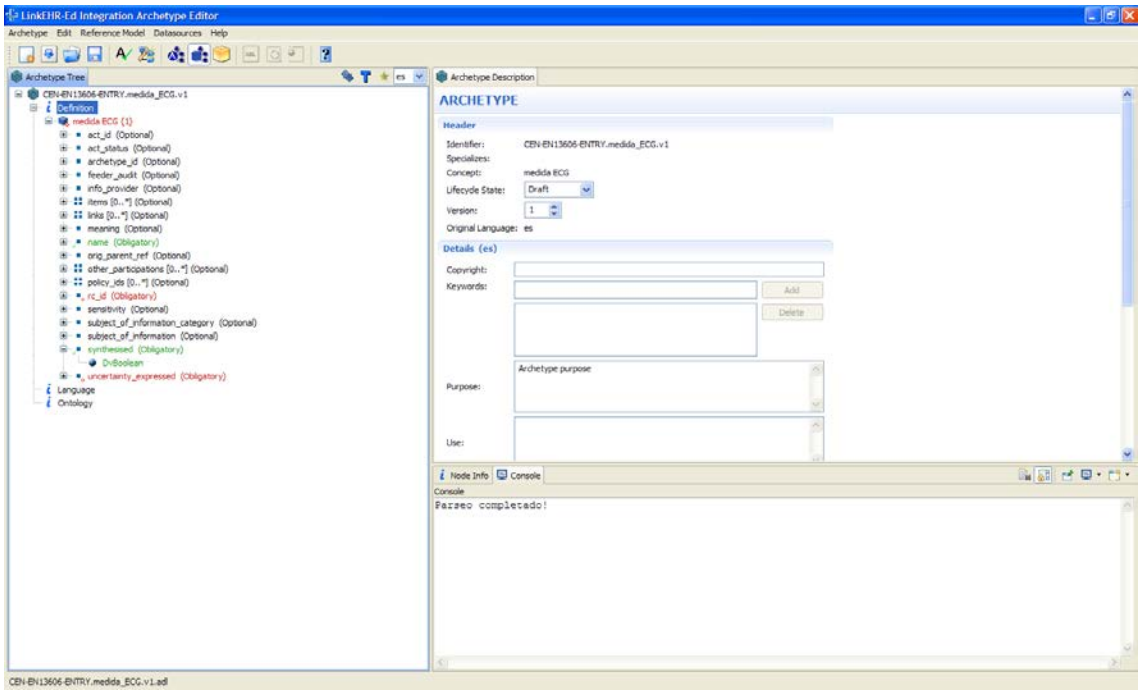


Figure 34. Mapping view

At the left side we can see the comprehensive archetype tree that has been automatically generated. Colours indicate the state of mappings:

- Black: This node does not need to be mapped because it or its son nodes are not mandatory, but the node can be mapped anyway if you wish.
- Red: This node is mandatory (including the full archetype path to reach it) and has not a defined mapping or a single value in the archetype definition.
- Green: This node is mandatory and has already been mapped.

If we click a leaf or atomic node (an integer, string, double, Boolean or date/time) the right side of the editor changes to show the mapping interface for that node (Figure 35).

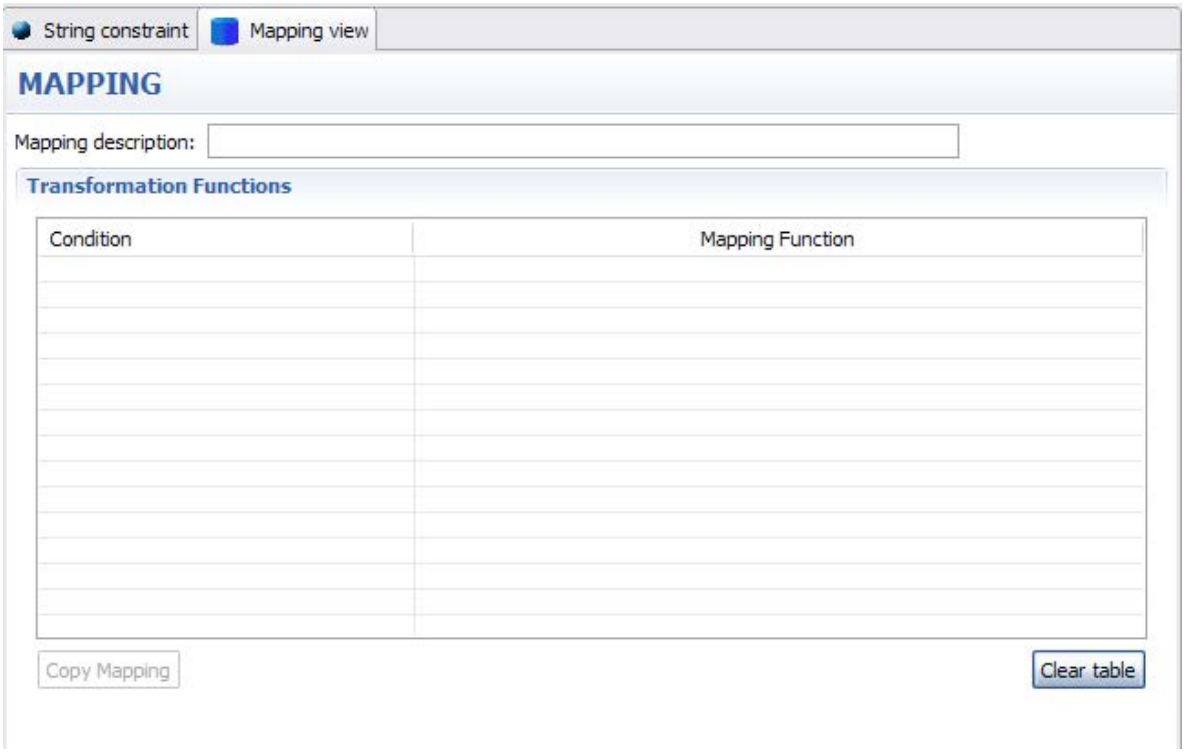
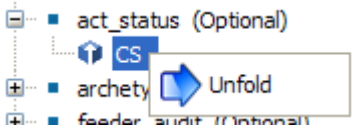


Figure 35. Mapping definition view

***NOTE**: If we want to map a node that inside an object defined as an Archetype internal reference, we can right-click that node and select the option* Unfold*. We can also double-click the internal reference node. This will create a clone of the referenced node to allow mapping it. The original archetype definition will not be affected by this action.*



Mapping view is divided in two main parts:

- First, there is a field to introduce a comment about the current mapping.
- Second, the section *Transformation functions* contains the table of mappings and transformations that are applied in order to get a value for the archetype attribute. This table has two columns, *Condition* and *Mapping Function*. The first one permit to specify filtering predicates which describe the conditions that source data must satisfy to be used in a transformation function. Note that the order is relevant and only the first applicable function is used. The

different rows of the table are evaluated consecutively, that is, first row condition is evaluated first, second condition is evaluated if first fails, and so on. It is recommended to add a 'default' *true* condition as the last one in order to assure that we actually get a result. If a condition is accomplished, then the transformation function is applied. Both conditions and functions use the same syntax and will be explained in following sections of this document.

5.3.1 Data source import

To import a data source means to import the original or legacy data schemas into LinkEHR in order to define the necessary mappings. These schemas are an XML Schema for XML data and a database schema for relational data.

***NOTE**: Currently LinkEHR only fully supports mappings to XML data sources. Relational data sources are supported as long as a canonical XML view is generated*

To add a new data source we only have to push the menu *Datasource* → *Datasource manager* and press *Add* on the manager dialog. The window that appears lets us to select the data source schema and to define an alias that will be used internally to reference to the source when defining mappings.

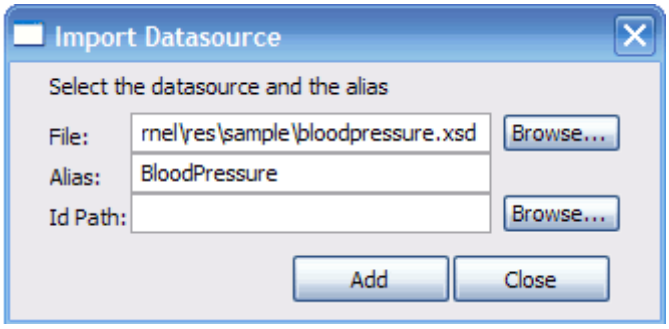


Figure 36. Window for the importation of a new data source

The third option (*Id Path*) is fundamental. By clicking the *Browse* button we will get a summary of the imported schema. We must select the field or element where the main identifier of data is located. When talking about health data, this identifier should correspond to the patient identifier. This information is necessary to avoid merging data from two different patients accidentally.

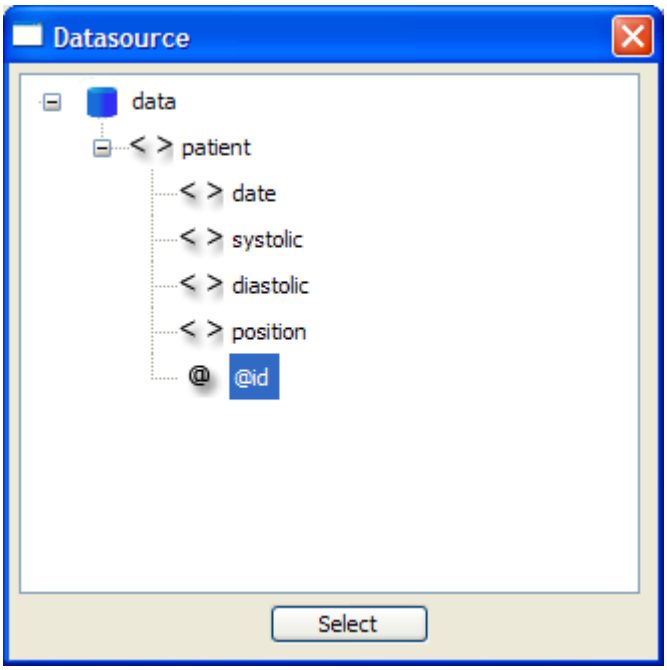


Figure 37. Selection of the patient identification entity

Datasource manager allows also the exportation and importation of data sources. e.g. if a data source is used by several people on different LinkEHR installations then only one needs to define the source alias and id path. This person can then *export* that source in order to the remaining people to *import* it.

5.3.2 Attribute mapping definition

An attribute mapping is defined as a set of pairs (filter, function). The filter defines the conditions that source data must satisfy to use the corresponding function. When a filter is missing it is supposed that its value is true; therefore it will be applied always. LinkEHR allows the definition of multiple filter-function pairs in the specification of a single value mapping. For instance, the following table contains a very simple value mapping transforming gender codes. Note that the order is relevant and only the first applicable function is used, consequently the last filter acts as a 'default' condition.

Filter	Function
/Patient/gender='M'	0
/Patient/gender='W'	1
True	9

Conditions and mapping functions that are defined at the transformation table use the same syntax. In LinkEHR a wide range of functions are available, apart from the typical arithmetic, boolean and string functions, it also supports time and date functions and archetype related functions. For instance, it includes functions for the transformation

of source time and date values into values conforming to the international standard ISO 8601 for date and time representation. Archetype related functions allow retrieving archetype metadata such as entity identification and linguistic entities defined in the ontology section like the text and description attached to archetype entities. By double-clicking the appropriate field of the table, the expressions editor will appear (Figures 38, 39, 40, 41, 42 and 43).

The main text field is where the expression is edited. It can be validated syntactically by clicking the button *Validate*. The different tabs of the window give access to the available operators and functions.

***NOTE**: String literals must be always double quoted. Parenthesis can be used anywhere in the expression. Equality operator is =. References to relational data source elements always begin with the symbol $. References to XML data sources are expressed on XML paths*

Operators tab

Here we can see the basic and common math and logical operators. They can be written directly in the expression, this tab is only a reference help.

IS, NOT and NULL are used to check data values of a data source. For example, a condition to check if the field Age from the table Person has a value would be written:

$Person.Age IS NOT NULL or either *NOT ($Person.Age IS NULL)*

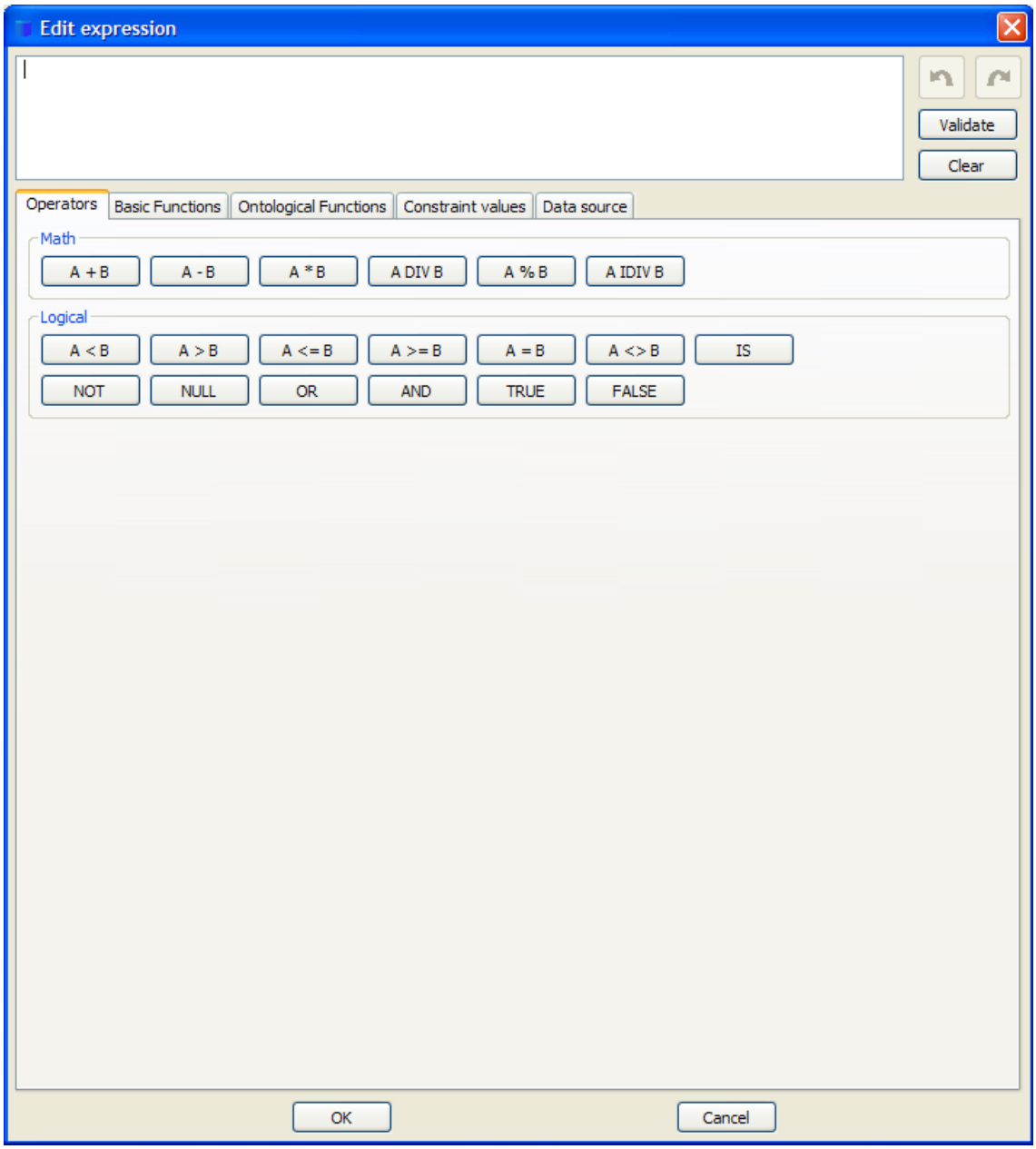


Figure 38. Logical and mathematical functions

Functions tab

Three types of functions are included here: Math functions, String functions and Date functions.

Math functions:

- **Abs**: Returns the absolute value from the expression.
- **Ceiling**: Turns a real number to its nearest greater Integer.
- **Floor**: Turns a real number to its nearest lesser Integer
- **Round**: Turns a real number to its lesser (≤0.5) or greater (>0.5) Integer.
- **Round half to even**: Rounds a real number with the required decimals. (e.g. @round-half-to-even(345.674,1) = 345.7. @round-half-to-even(345.674,2) = 345.67)

- **ToFloat**: Transforms a Number or String to a Float.
- **ToInteger**: Transforms a Number or String to an Integer.

String functions:

- **Left**: Trims a string from the left a number of characters.
- **Right**: Trims a string from the right a number of characters.
- **Starts with**: Returns if a string starts with another string.
- **Ends with**: Returns if a string ends with another string.
- **Lowercase**: Returns the lowercase string of the string.
- **Uppercase**: Returns the uppercase string of the string.
- **Contains**: Returns if a string contains another string.
- **Matches**: Returns if a string matches a regular expression.
- **Concat**: Concatenates two strings (resulting in another string).
- **Normalize space**: Returns the string without all the spaces and indentation characters.
- **Subsequence**: Returns the string that is the subsequence of the string from a position.
- **String length**: Returns the length of the string.
- t**oString**: Transforms parameter to a string.

Date and time functions:

- **toISODate/Time():** transform a date or time value from the original source into the ISO format. A regular expression with the syntax of the original date/time must be provided.
- **Years/months/days/hours/minutes/seconds from Date/Time**: Extracts desired value from an ISO Date/Time
- **Current Time**: Functions to obtain current system time.

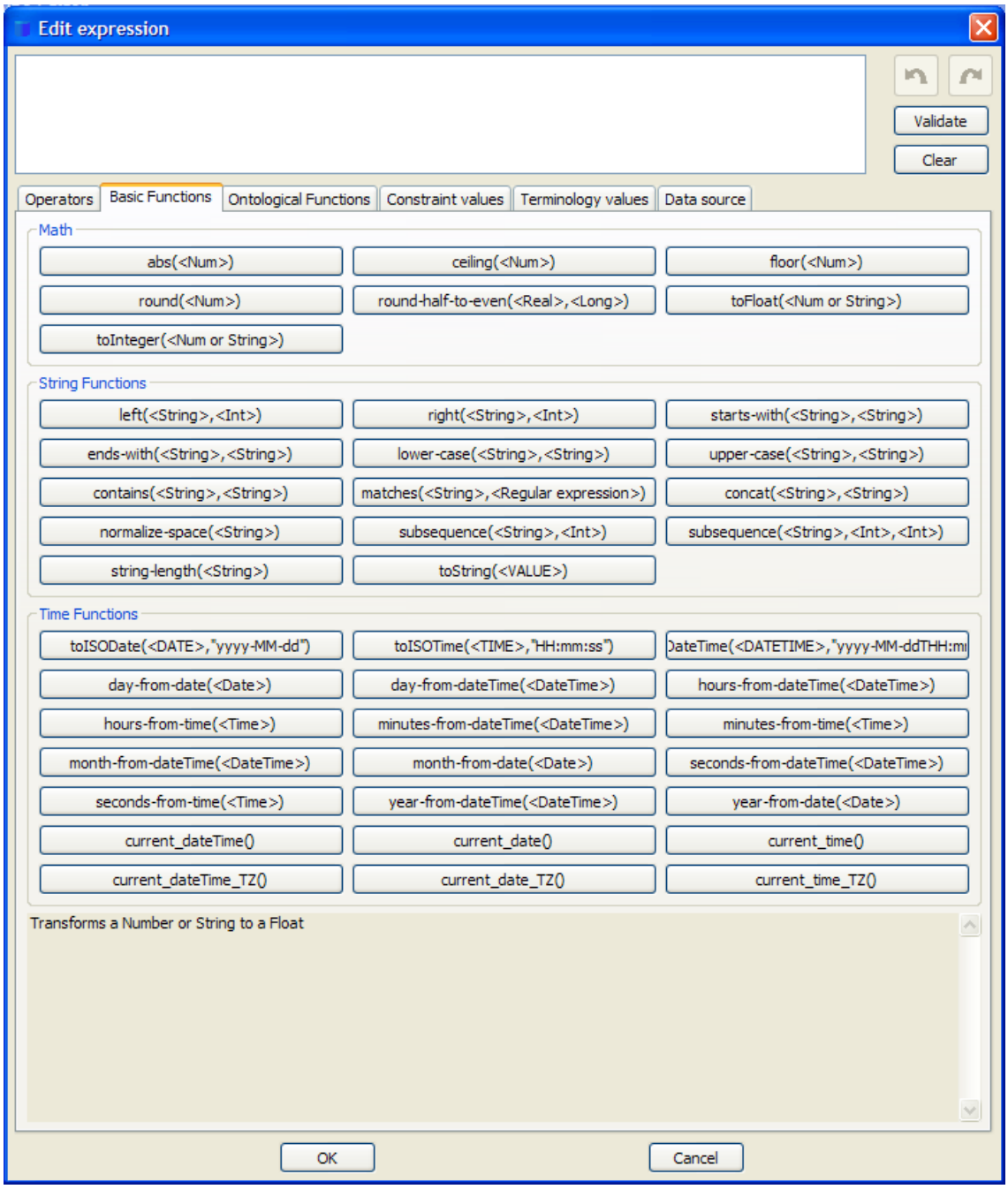


Figure 39. Basic functions provided by LinkEHR

Ontological functions tab

Some utility functions and functions to access to the ontological information of the archetype are provided.

- **id():** this function calculates a unique identifier for data instances.
- **code():** extracts the atXXXX code of parent C_OBJECT.
- **text():** extracts the corresponding text associated with the atXXXX code and language.

- **description():** extracts the corresponding description associated with the atXXXX code and language.

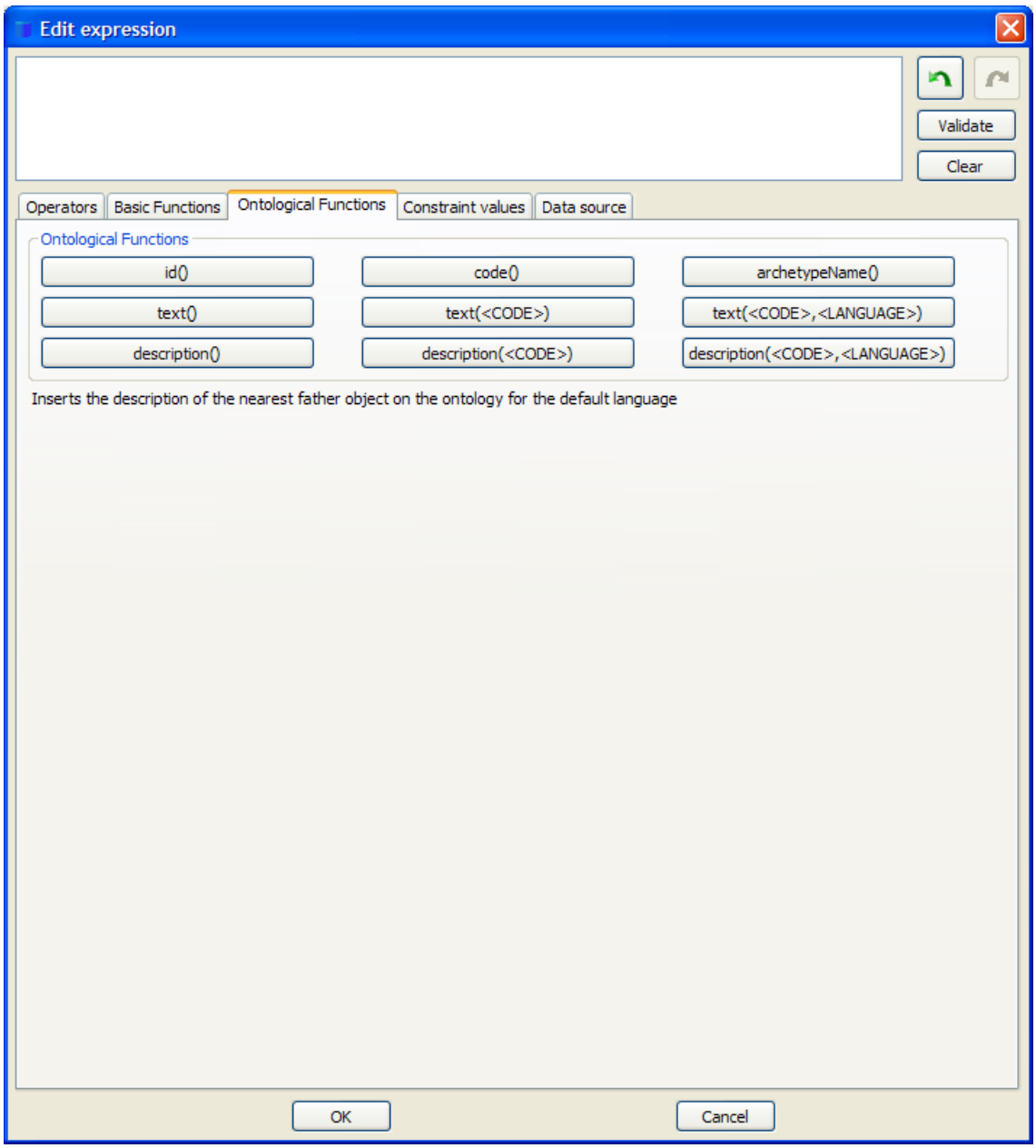


Figure 40. Ontological functions provided by LinkEHR

Constraint values tab

Constraint values tab gives a direct access to the list of allowed values of a primitive object. Since this list indicates the only valid values for the archetype attribute, one of them must be chosen as the result value for the expression.

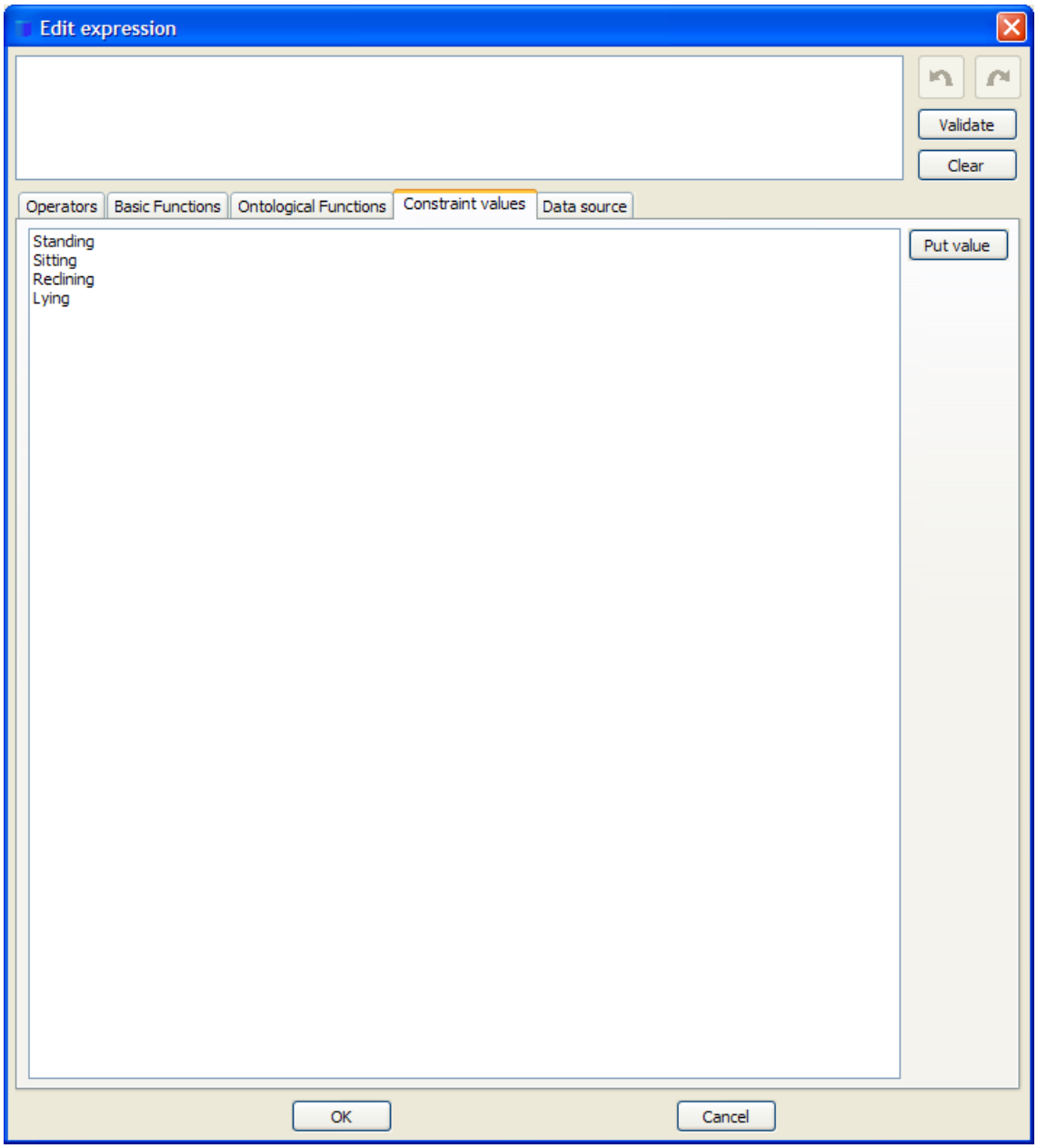


Figure 41. Constraint value tab

Terminology values tab

Terminology values tab gives a direct access to the values allowed on that node. Those values are extracted from the terminology defined for each attribute on the documentation window (check “Defining a reference model documentation”). For each one of the items on the list, either the text or the code can be inserted on the expression.

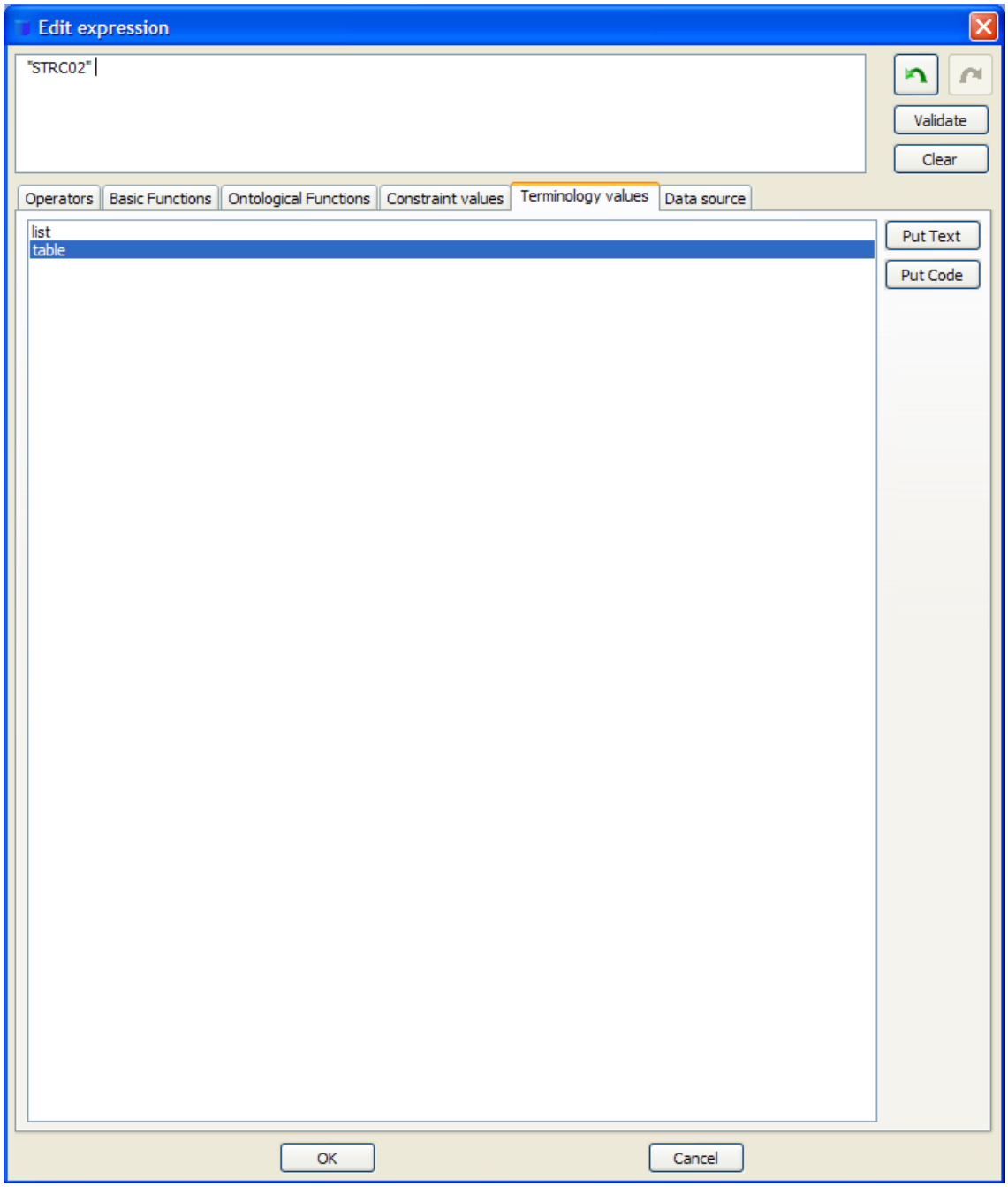


Figure 42. Selection of a value from a set of valid values as stated in the documentation attached to the reference model.

Data source tab

Maybe the most important tab, it shows a view of the data source schema that was selected when creating this mapping entry. We just have to select the field we want to use in a expression and include it. We can add several fields and operate with all them. They will act as variables from where data will be extracted.

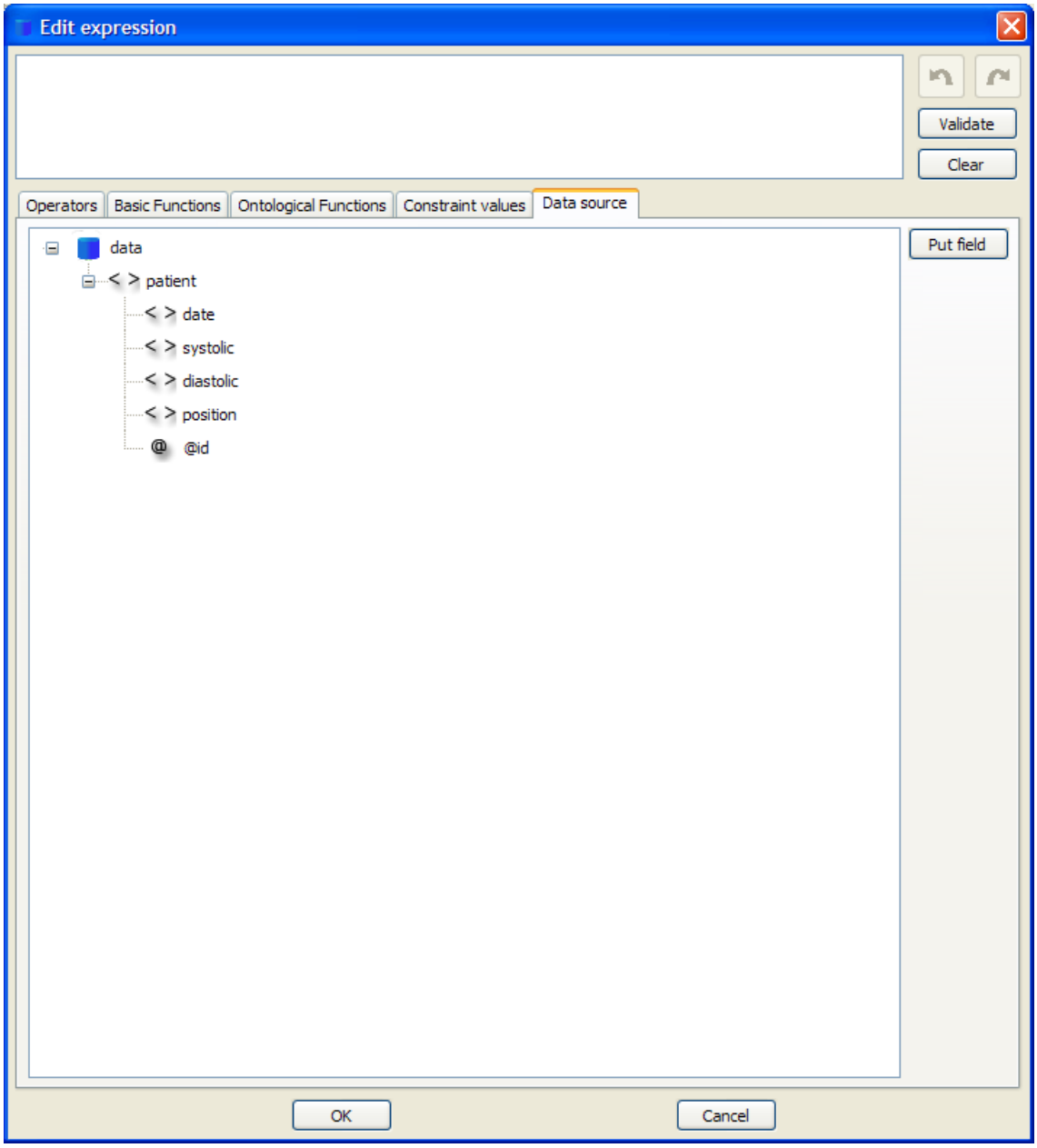


Figure 43. The display of a simple data source

***NOTE**: LinkEHR mapping editor has also a coding assist feature to ease the textual mapping definition*

5.3.3 Object builder definition

In addition to attribute mappings, object builder can be attached to archetype node objects with multiple occurrences. An object builder comprises a source path and a condition. Object builders control the creation of target instances, in such a way that a new target instance is constructed for each source element addressed by the path that satisfies the filter. When no object builder is defined default semantics is applied, which groups together instances that share the same context following a minimum-cardinality approach, i.e. it is just generated the minimum number of elements necessary for the result to be compliant with the archetype. Variables can be created in order to reuse common data sources paths, as seen in Figure 44.

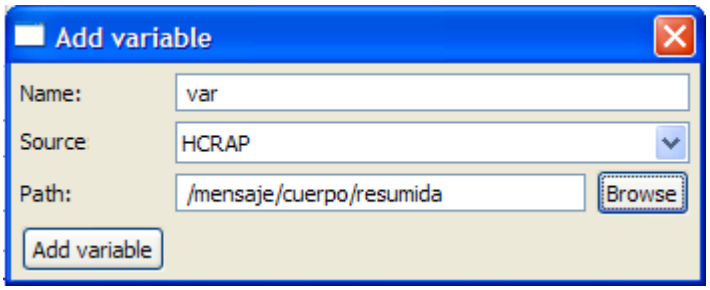


Figure 44. Adding a variable

The condition expression is edited with the same editor as the attribute mappings (see "Attribute mapping definition") with the only difference of the data source paths, which are now dependant of the defined variables.

On this window is also possible to say if an object mapping has bag semantics and which is the parent object mapping of this object mapping (mapping without target paths can be parents of an object mapping, which allow reuse of object mappings).

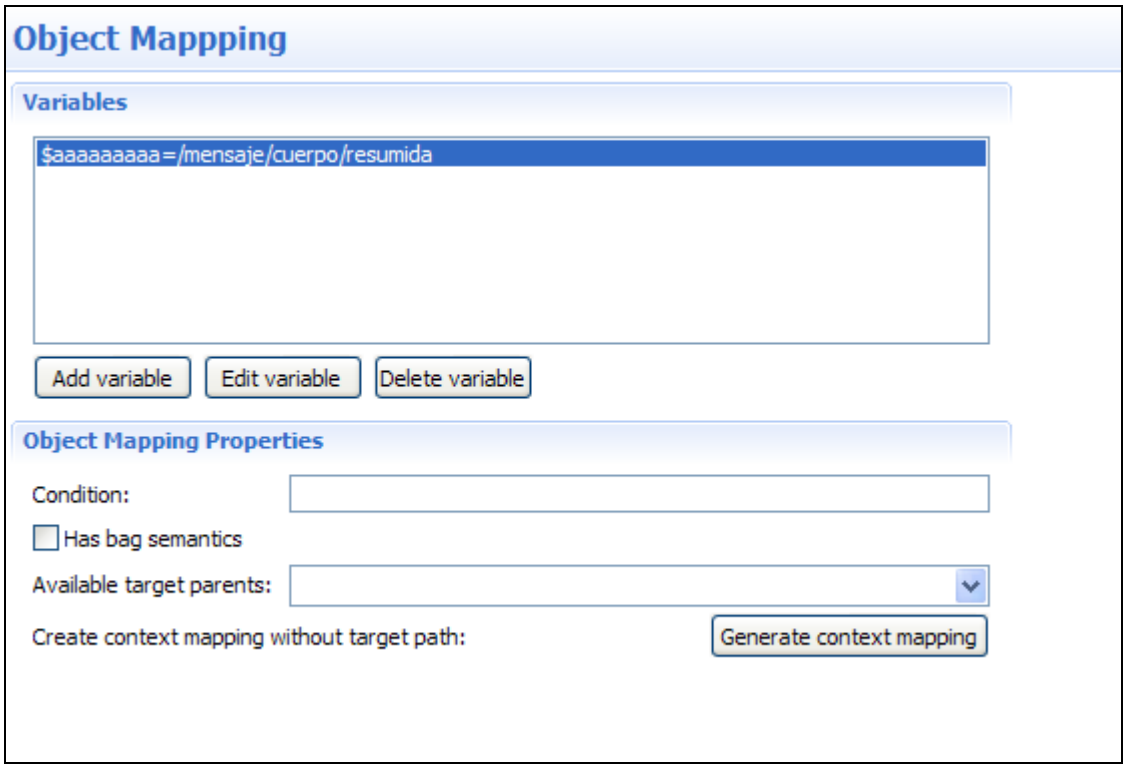


Figure 45. Object mapping window

5.3.4 Generation of covers and XQuery transformation scripts

Once we have defined mapping at least for all mandatory paths (the root of the archetype must be coloured in green) we can generate automatically the transformation script by clicking the button of *Export Integration Archetype*.

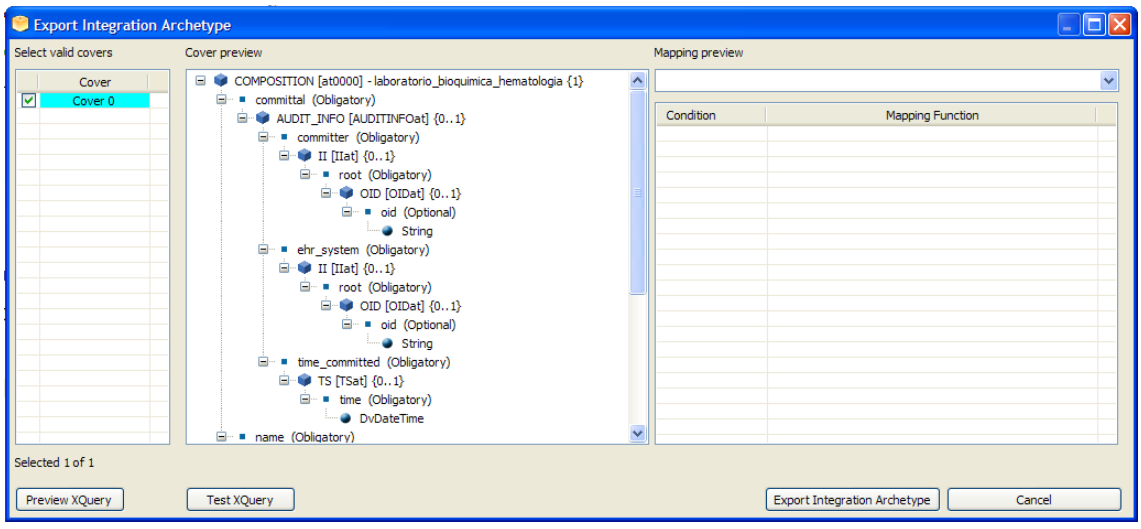


Figure 46. Export integration archetype window

***NOTE**: To save your mappings you must save the archetype explicitly by using the menu* Archetype→Save→Save Archetype *or the* Save Archetype button*. Mappings are saved in a file named as the archetype but with the .MAP extension that must reside in the same folder of the archetype .ADL file.*

The export integration archetype window includes several parts and options.

- Valid covers: A *cover archetype* is a valid combination of attribute mappings that can generate instances of data. In other words, when we define mappings for different data sources in the same archetype, different covers will be generated (one for each possible combination of sources). We can review these covers and select the ones we consider valid for our purposes. If several covers are left selected, a different XQuery will be generated for each one, but the application of these different queries will always return valid data instances with regard to the archetype and reference model definition. If only one data source is used in the mappings, only one cover will appear and must be left selected.

- Preview XQuery (Figure 47): Shows a window with the generated XQuery code that can be copied and used elsewhere.

- Test XQuery (Figure 48): We can select a test XML data file (which must be an instance of the data source schema used for defining the queries) and execute the generated query in order to see the resultant normalized XML document (Figure 49).

- Export Integration Archetype: This option generates an .LKR file which includes the generated scripts for the integration archetype.

***NOTE**: The result of using LinkEHR is the generation of* .LKR *files. These files contain all the necessary information to access, extract and transform legacy data into archetyped data instances; and will be the input for LinkEHR Integration Engine. This module is a data server which serves normalized instances of data or standard EHR extracts from the original data sources.*

The generated transformation script can be used by any Information System with XQuery execution capabilities. LinkEHR uses the open source engine Saxon-B from http://www.saxonica.com/

Please note that some transformation functions used at the generated XQuery will require the use of third-party libraries, such as the Joda-Time library for ISO dates manipulation. http://joda-time.sourceforge.net/

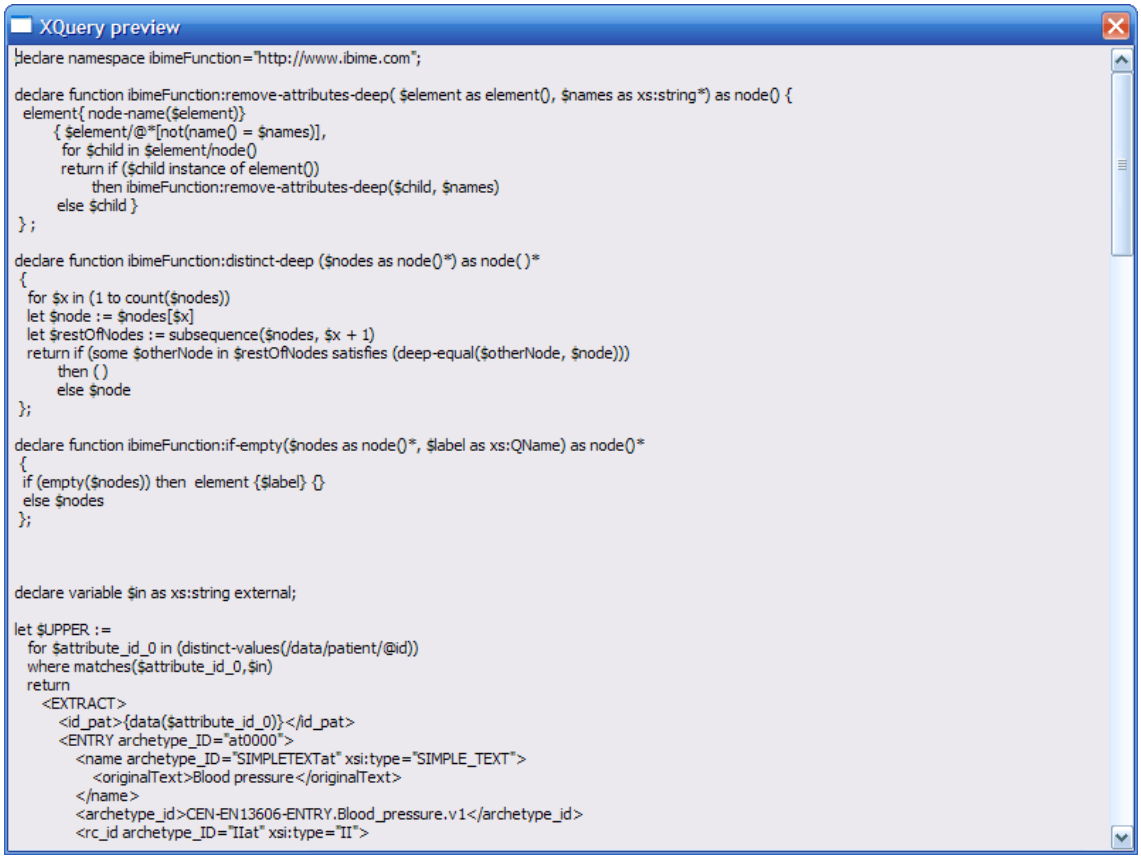


Figure 47. Preview of the generated XQuery

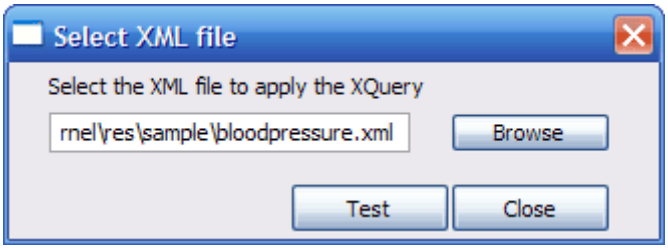


Figure 48. Testing an integration archetype over a data XML

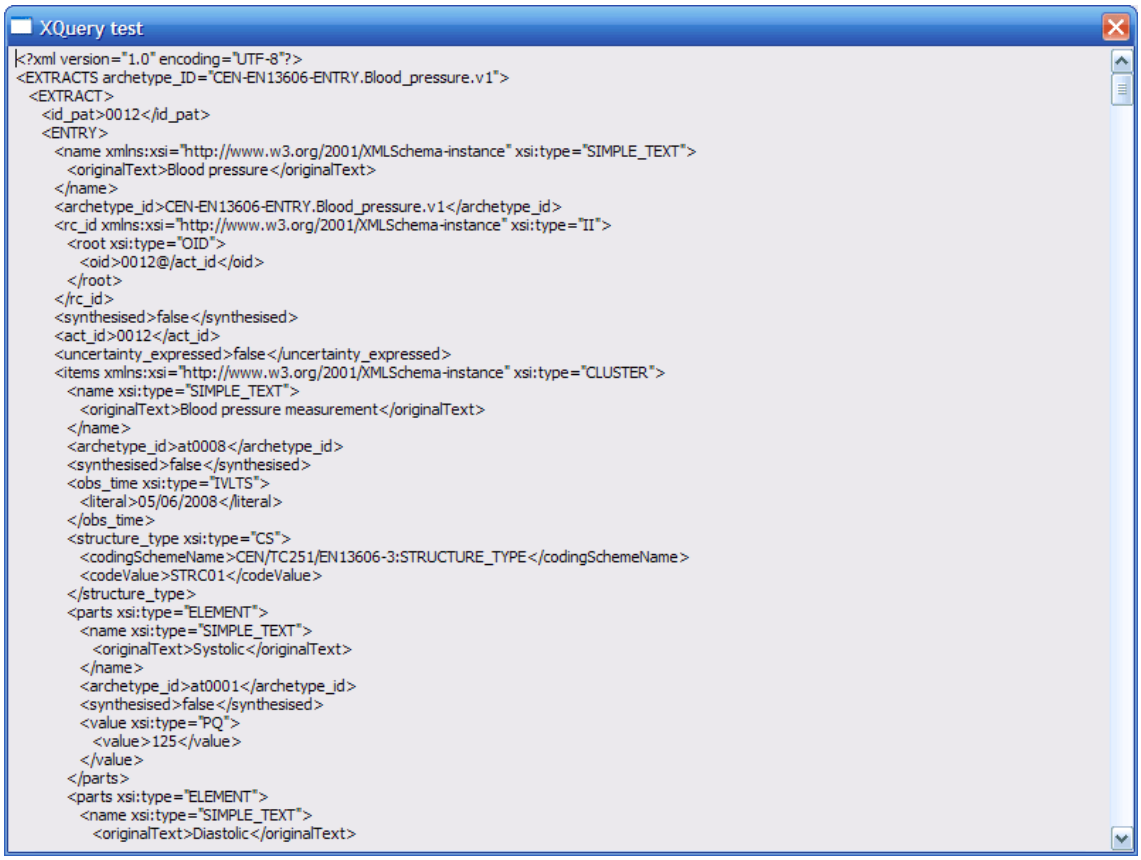


Figure 49. Normalized XML which results of the application of the integration archetype

COM

6. Customized archetype editor for CEN EN13606

To activate CEN EN13606 editor you must activate the option “Use specific editor when available” under the Visualization tab on LinkEHR configuration dialog (see Figure 4). After doing this, whenever you create or edit a CEN EN13606 archetype the specific editor will also be opened.

6.1 Choosing a root entity for a CEN EN13606 archetype

Before starting the proper archetype edition, it is necessary to choose which will be the root entity of your archetype. CEN EN13606 provides six different entities:

- FOLDER: A level of organization used to group information on clinical episodes, problems, time periods or clinical service
- COMPOSITION: A complete clinical document in the scope of a clinical session.
- SECTION: EHR data that are inside of a composition and comes from a single Figure process
- ENTRY: A clinical fact that is registered.
- CLUSTER: Groups data into lists or tables.
- ELEMENT: Container of the leaf nodes of an EHR structure.

Usually, root class is a COMPOSITION or ENTRY. Figure 50 shows the archetype class selection algorithm for CEN EN13606

Archetype definition class selection algorithm for CEN EN13606

Adapted from openEHR

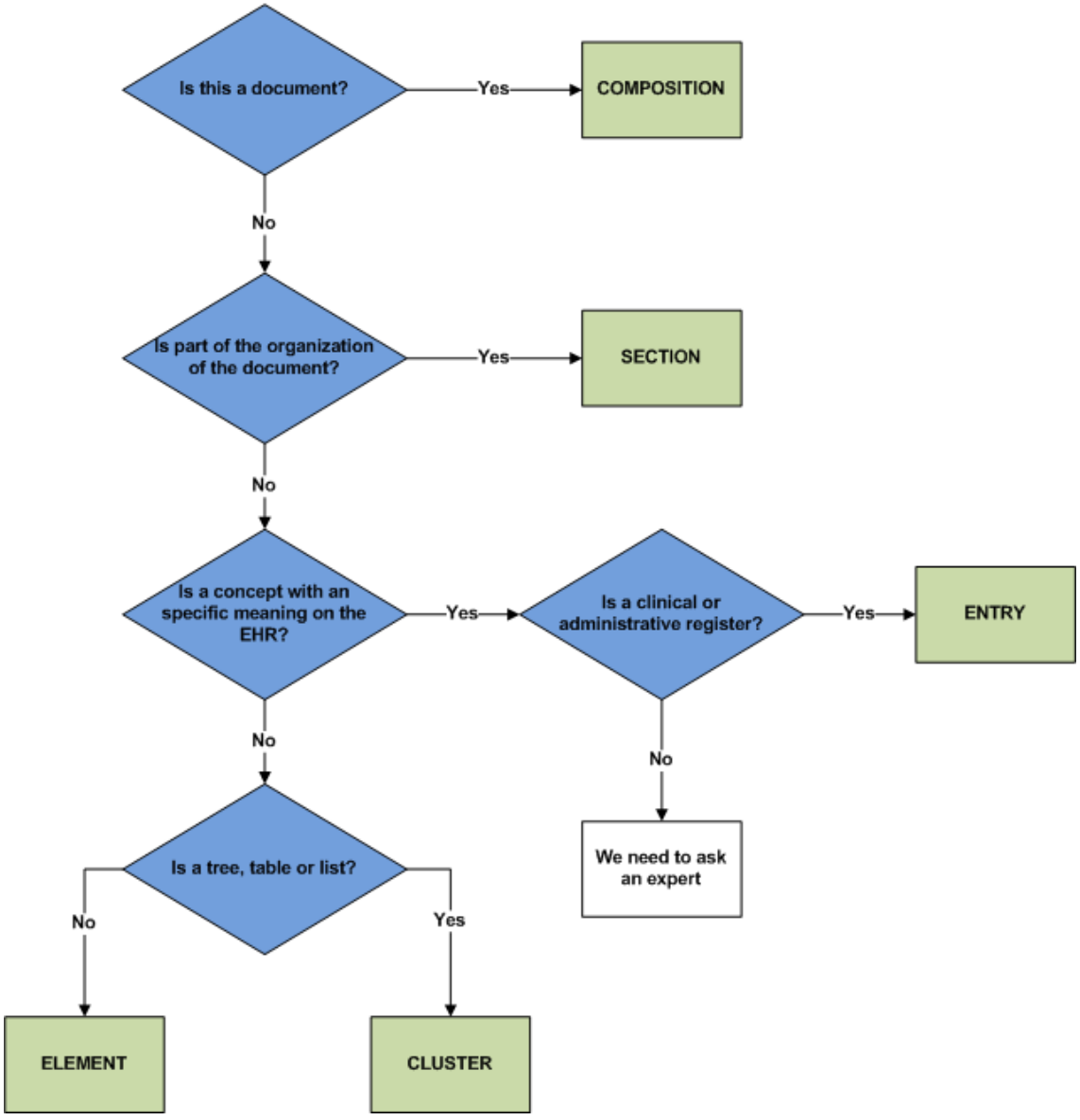


Figure 50. CEN EN13606 class selection algorithm

6.2 Editing CEN EN13606 archetypes with LinkEHR

By creating a new archetype (selecting '*CEN*' as organization and '*EN13606*' as reference model) the archetype tree view will look like this (here we have chosen a COMPOSITION with name "document test", note that root icon will change if a different root entity is chosen).

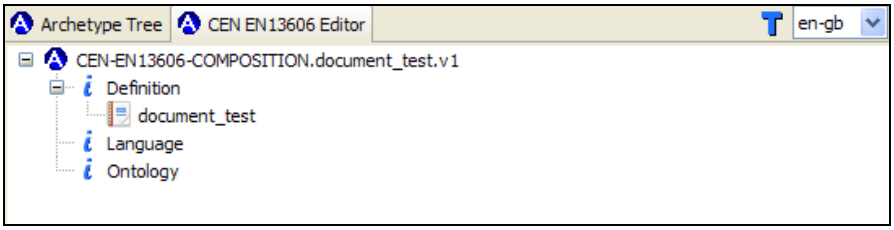


Figure 51: A newly created composition

From there, clicking on a node shows its properties. Note that "*ANY ALLOWED*" is checked, so you must uncheck it to change its properties right away (but if you keep editing the archetype tree, LinkEHR will take care of this for you).

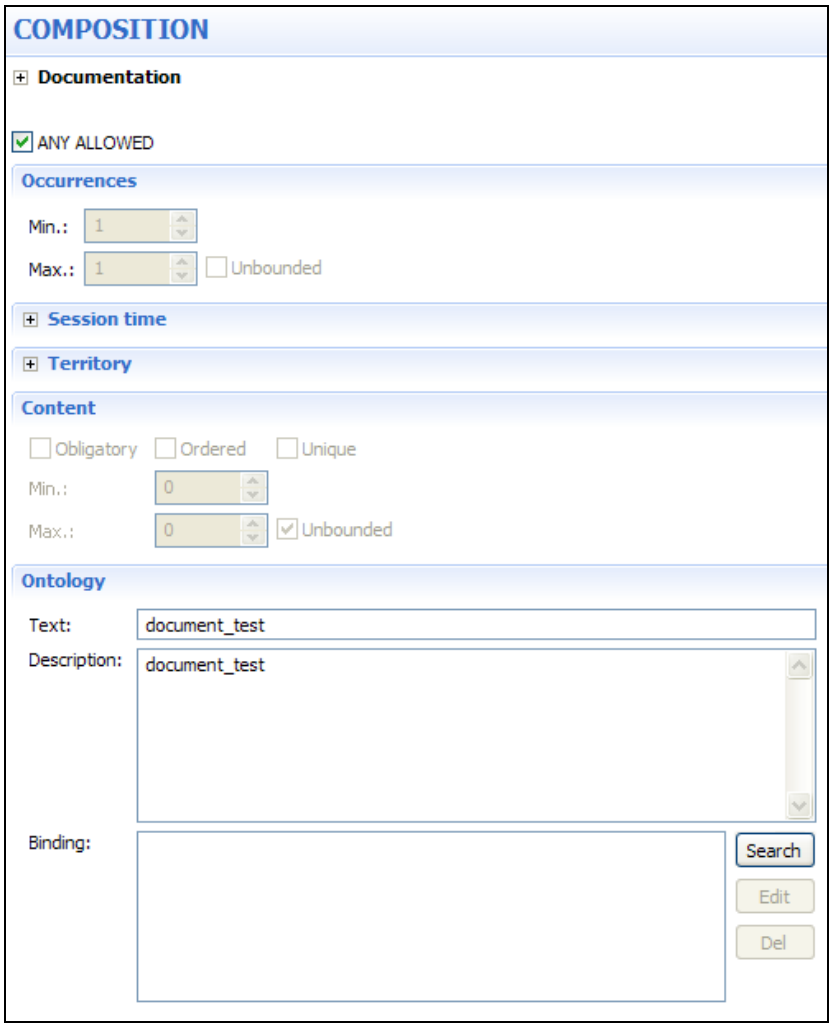


Figure 52: COMPOSITION properties form

On CEN EN13606 editor, each one of the reference model classes has its own view to change its properties. Every view has three things in common: The "*ANY ALLOWED*" (if this checkbox is checked means that no further restrictions will be applied to this node of the archetype), the "*Occurrences*", which mean how many distinct objects of this kind will appear on the data, (e.g. A patient can have one or two different telephone numbers so one 'telephone number' node can be created with minimum occurrences of one and maximum occurrences of two will be enough to express that restriction), and lastly the "*Ontology*" section, where a text, a description and binding can be set in order to make richer archetypes.

Each one of CEN EN13606 classes has also a part of the form to change how many children can have. On FOLDER class it is called *sub-folders*, on COMPOSITION class it is called *content*, on SECTION class it is called *members*, on ENTRY class it is called *items* and on CLUSTER class it is called *parts*.

Other additional properties can be changed for each one of the classes available on the reference model.

Next step is creating a child for the root object. This is done for the node menu by either right clicking on a node or pressing the "Node…" button on the toolbar. This will pop up a menu like the one in Figure 53.

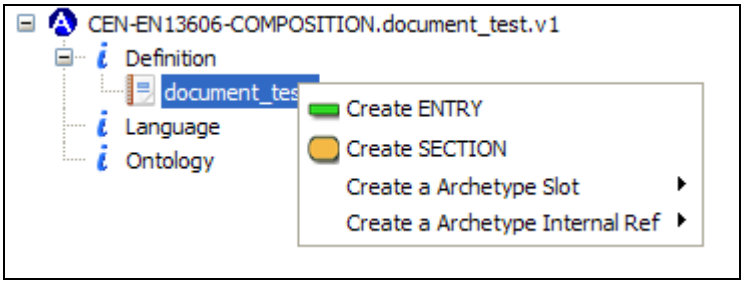


Figure 53: Creating a new object on the tree

Selecting one of the alternatives will add this node to the archetype tree.

6.2.1 CEN EN13606 Datatypes

Datatypes are the leaf nodes of the archetype tree on CEN EN13606 editor. Each one of the data types is intended to be used for a specific purpose.

- BL: Boolean data value. Value data type that can be true or false.
- CODED_TEXT: A free text string with an associated coded value.
- CV: Coded Value. Coded data, stating only a code, without classifiers or translations to other codification systems.
- DATE: Identifies a single day of calendar, expressed by a combination of calendar year, calendar month, calendar week, calendar day or day of the year.

- DURATION: A period of time from a non-fixed set of time (which is not specified). It can be expressed as a negative duration, meaning that duration is backwards. It must not be used to express points of time (for this, use TS)

- ED: Encapsulated data. Data which main purpose is beign interpreted by humans. This data includes any written language, multimedia data, digital signatures or information defined in any other standard.

- IVL: Interval. An interval in CEN EN13606 can contain PQ, TS, DURATION, ORD and RTO. It is defined with a start and end object of the selected data type. As is widely used, the interval of time has its own type (IVLTS).

- IVLTS: Interval time. Can be defined by a start date or time and an end date or time, by a duration without start or end, by a start date or time and a duration, and by a duration and an end date or time. Do not confuse with DURATION

- ORD: Ordinal. A number that defines a position in a list or series with a textual description.

- PQ: Physical quantity. A dimensioned quantity that expresses the result of a measurement.

- RTO: Ratio. A quantity built as the quotient of a numerator quantity divided by a denominator quantity.

- SIMPLE_TEXT: A simple text without an associated code.

- TS: Time point. A non-dimensional time moment.

- URI: Universal resource identifier. A telecom address as specified in standard RFC 1738

6.2.2 Creating archetype internal references and archetype slots

To avoid unnecessary repetition of structures and ease the reutilization of archetypes, archetypes have two kinds of nodes that allow this.

Archetype internal references

To ease internal structure reuse, archetypes have special nodes called “Archetype internal references”. The meaning of these nodes is that the structure for them is exactly the same as the one this is targeting to.

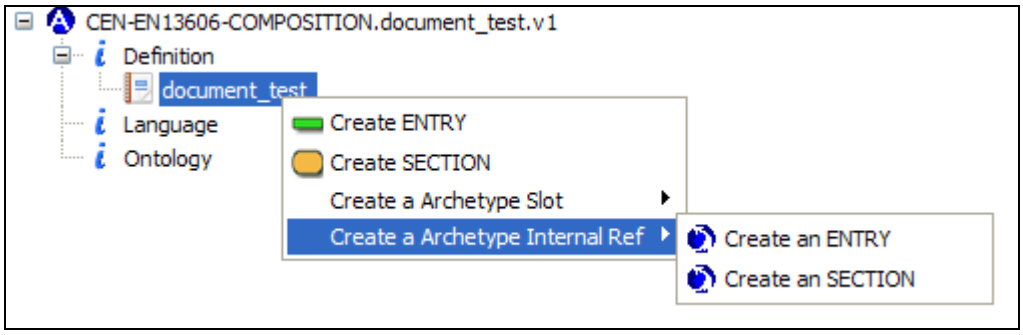


Figure 54: Creating an archetype internal reference

When selecting the archetype internal reference you want, LinkEHR shows a list of the archetype tree nodes that where the internal reference can point.

NOTE: *You can not create an archetype internal reference to a sibling node. If you need more than one of the same object, change occurrences accordingly.*

Archetype slot

In addition to internal references, you can define points on the archetype (called archetype slot) where you can include external archetypes. Archetype slots have the option to include and to exclude a pattern of archetypes.

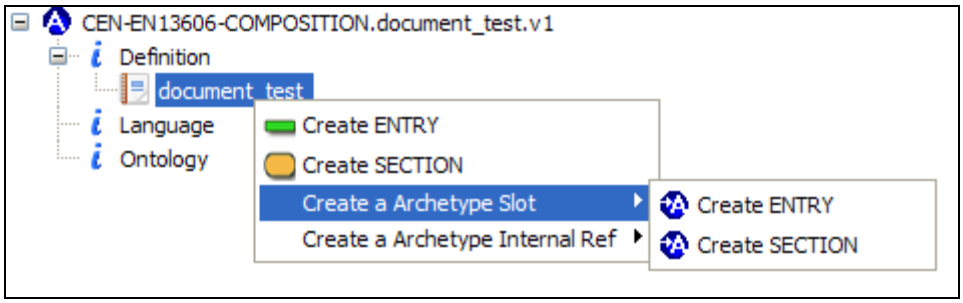


Figure 55: Creating an archetype slot

Figure 56 shows the slot properties view. Note that multiple includes and excludes can be defined. In that case, “edit includes” and “edit excludes” allow the definition of the different conditions.

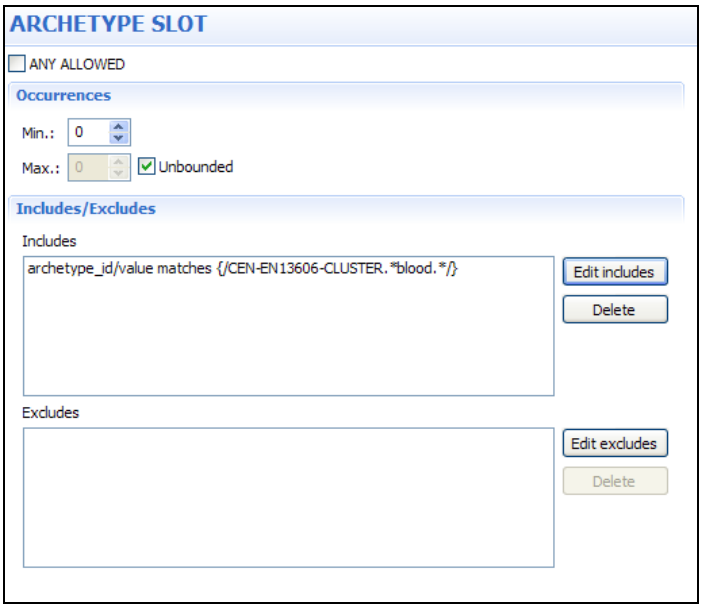


Figure 56: Archetype slot properties view

6.3 Editing HL7 CDA archetypes with LinkEHR

6.3.1 Choosing a root entity for a HL7 CDA archetype

Before starting the proper archetype edition, it is necessary to choose which will be the root entity of your archetype. HL7 CDA provides six different entities:

- CLINICALDOCUMENT: this element typically contains all of the namespace declarations necessary for the clinical document; may declare the realm for which the document which was written; always indicates the version of CDA in use; and may declare the business rules that this document asserts conformance to.
- AUTHOR: represent the first of three different kinds of information sources from the HL7 RIM that appear in the CDA, and is arguably the most importance since there must be at least one author in a CDA instance.
- CUSTODIAN: the custodian association links the assigned custodian to the clinical document. The assigned custodian is the organization that has been assigned the role to be the steward of the clinical document.
- PARTICIPANT: this kind of participant allows CDA to record associate other roles with a clinical document to support use cases not originally anticipated.
- SECTION: EHR data that are inside of a composition and comes from a single Figure process
- ENTRY: A clinical fact that is registered.
- ACT: represent any kind of clinical act.
- PROCEDURE: this class is also very much like an act. It's an act whose outcome results in the physical alteration of the subject.
- ENCOUNTER: is used to describe acts that are clinical encounter between a healthcare provider and a patient.
- OBSERVATION: is very similar to the procedure class and can be thought of as a "non-altering" procedure that results in a value.
- OBSERVATIONMEDIA: is a restricted form of the observation class intended to support other forms of media, such as an imaging result.
- ORGANIZER: is a specialization of the act class that is designed to support grouping of information.
- SUBSTANCEADMINISTRATION: is very similar to the procedure class, but has additional class attributes and associations to address medication specific information.
- SUPPLY: this class is used to describe things that are given to the patient for their subsequent use, possibly, as in the case of medications, for later administration.

Usually, root class is a CLINICALDOCUMENT. Figure 1 shows the dialog form to select the root element.

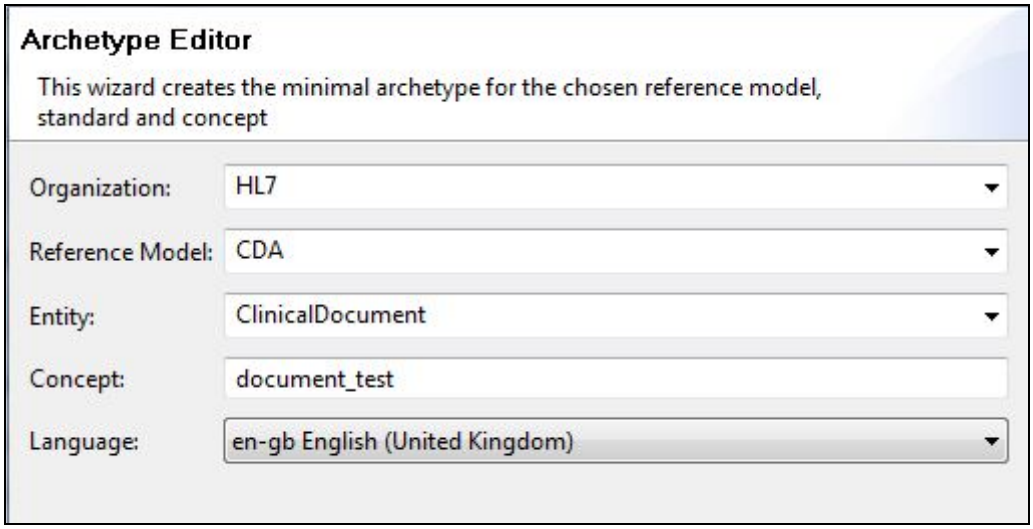


Figure 57. Selecting a CDA document.

6.3.2 Editing HL7 CDA archetypes with LinkEHR

By creating a new archetype (selecting '*HL7*' as organization and '*CDA*' as reference model) the archetype tree view will look like this (here we have chosen a ClinicalDocument with name "document_test", note that root icon will change if a different root entity is chosen).

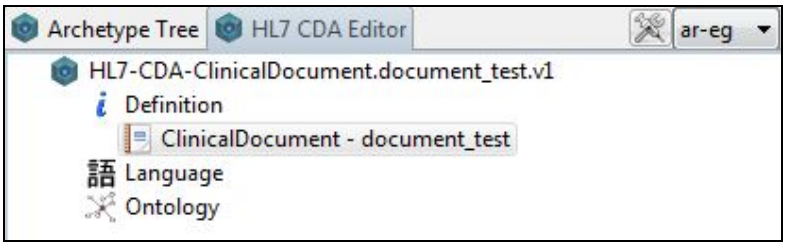


Figure 58: A newly created ClinicalDocument

From there, clicking on a node shows its properties. Note that "*ANY ALLOWED*" is checked, so you must uncheck it to change its properties right away (but if you keep editing the archetype tree, LinkEHR will take care of this for you).

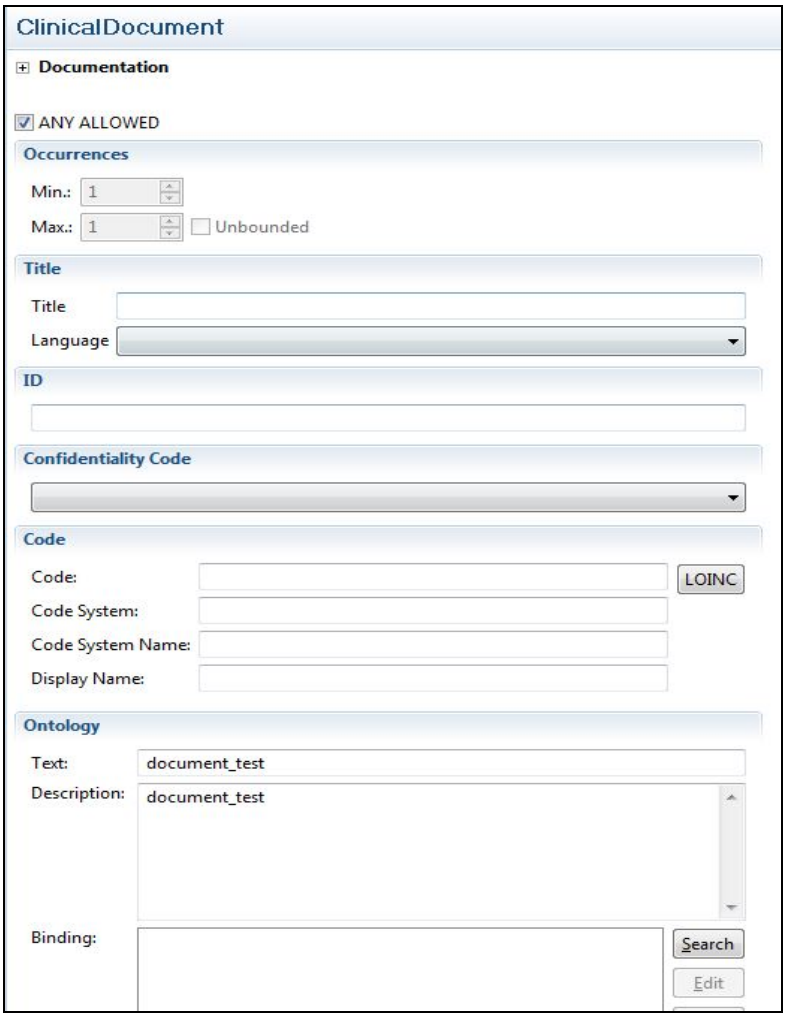


Figure 59: ClinicalDocument properties form

On HL7 CDA editor, each one of the reference model classes has its own view to change its properties. Every view has three things in common: The "*ANY ALLOWED*" (if this checkbox is checked means that no further restrictions will be applied to this node of the archetype), the "*Occurrences*", which mean how many distinct objects of this kind will appear on the data, (e.g. A clinic act can have one or two different observations, so one *observation* node can be created with minimum occurrences of one and maximum occurrences of two will be enough to express that restriction), and lastly the "*Ontology*" section, where a text, a description and binding can be set in order to make richer archetypes.

Other additional properties can be changed for each one of the classes available on the reference model.

Next step is creating a child for the root object. This is done for the node menu by either right clicking on a node or pressing the "Node…" button on the toolbar. This will pop up a menu like the one in Figure 53.

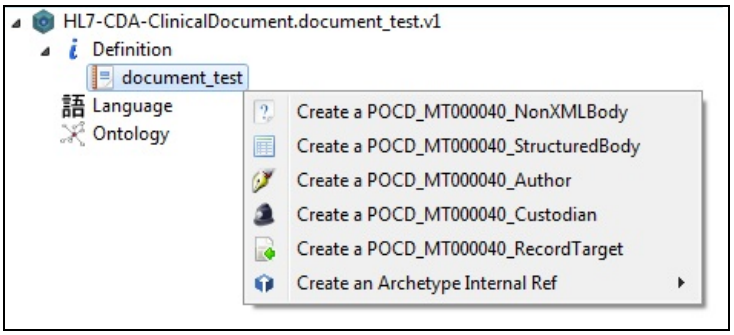


Figure 60: Creating a new object on the tree

Selecting one of the alternatives will add this node to the archetype tree.

6.3.3 HL7 CDA Datatypes

Datatypes are the leaf nodes of the archetype tree on HL7 CDA editor. Each one of the data types is intended to be used for a specific purpose.

- BL: Boolean data value. Value data type that can be true or false.
- INT: Integer data represents a positive or negative integer or zero. Negative integers are preceded with a minus sign. The integer (INT) data type is used sparingly in the CDA schema.
- REAL: the real data type follows from INT, save that it contains positive or negative real numbers or zero, rather than just integers. Appear in the value attribute as either decimal numbers or double precision floating point numbers in XML.
- ST: the String data is perhaps the easiest to understand. It encodes simple text data.
- CD: the Concept Descriptor, in addition to several attributes describing the code, it can contain a reference to the original text.
- CE: the Coded with Equivalents type is used to exchange coded concepts that are not permitted to contain qualifiers and so do not allow for codes to be created compositionally using post-coordination.
- CS: the Coded Simple type is used to convey codes that have a fixed value for *codeSystem.*
- CV: the Coded Value data type derives from the CE data type. It is used when only an unqualified coded concept without translations is desired in an exchange.
- II: the Instance Identifier data type is used to identify different instances of a kind of thing. It is used extensively in the CDA specification to identify persons, places, things, actions, roles, etc.
- ED: Encapsulated data is the way that HL7 transmits data in formats not defined by HL7. This data can include images, video, audio, etc. The data may

be referenced, or the data may be directly incorporated into the CDA document.

- IVL_TS: Interval of time data is often used to record a time interval over which some observation or event occurred or is intended to occur. Intevals of time are be specified using at most any two of the following components.
- TS: a time stamp is an instant time. Since this is implicitly a quantity of time since some arbitrarily chosen epoch of time, it is part of the quantity hierarchy. The HL7 standard does not define an epoch date to be used since a system can use any epoch value and still process time stamps correctly.
- PQ: Physical quantity. A dimensioned quantity that expresses the result of a measurement.
- RTO: Ratio. A quantity built as the quotient of a numerator quantity divided by a denominator quantity.
- URL: Universal Resource Locator. A telecom address as specified in standard RFC 1738